

Package ‘geoflow’

December 12, 2025

Version 1.1.0

Date 2025-12-12

Title Orchestrate Geospatial (Meta)Data Management Workflows and Manage FAIR Services

Description An engine to facilitate the orchestration and execution of metadata-driven data management workflows, in compliance with 'FAIR' (Findable, Accessible, Interoperable and Reusable) data management principles. By means of a pivot metadata model, relying on the 'DublinCore' standard (<<https://dublincore.org/>>), a unique source of metadata can be used to operate multiple and inter-connected data management actions. Users can also customise their own workflows by creating specific actions but the library comes with a set of native actions targeting common geographic information and data management, in particular actions oriented to the publication on the web of metadata and data resources to provide standard discovery and access services. At first, default actions of the library were meant to focus on providing turn-key actions for geospatial (meta)data: 1) by creating manage geospatial (meta)data complying with 'ISO/TC211' (<<https://committee.iso.org/home/tc211>>) and 'OGC' (<<https://www.ogc.org/standards/>>) geographic information standards (eg 19115/19119/19110/19139) and related best practices (eg. 'INSPIRE'); and 2) by facilitating extraction, reading and publishing of standard geospatial (meta)data within widely used software that compound a Spatial Data Infrastructure ('SDI'), including spatial databases (eg. 'Post-GIS'), metadata catalogues (eg. 'GeoNetwork', 'CSW' servers), data servers (eg. 'GeoServer'). The library was then extended to actions for other domains: 1) biodiversity (meta)data standard management including handling of 'EML' metadata, and their management with 'DataOne' servers, 2) in situ sensors, remote sensing and model outputs (meta)data standard management by handling part of 'CF' conventions, 'NetCDF' data format and 'OPeNDAP' access protocol, and their management with 'Thredds' servers, 3) generic / domain agnostic (meta)data standard managers ('DublinCore', 'DataCite'), to facilitate the publication of data within (meta)data repositories such as 'Zenodo' (<<https://zenodo.org/>>) or DataVerse (<<https://dataverse.org/>>). The execution of several actions will then allow to cross-reference (meta)data resources in each action performed, offering a way to bind resources

between each other (eg. reference 'Zenodo' 'DOI' in 'GeoNetwork'/'GeoServer' meta-data, or vice versa reference 'GeoNetwork'/'GeoServer' links in 'Zenodo' or 'EML' meta-data). The use of standardized configuration files ('JSON' or 'YAML' formats) allow fully reproducible workflows to facilitate the work of data and information managers.

Maintainer Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Depends R (>= 3.3)

Imports R6, methods, dotenv, benchmarkme, httr, mime, jsonlite, yaml, XML, xml2, rdfliib, curl, whisker, digest, dplyr, readr, arrow, zip, png, uuid, sf, sfarrow, terra, geometa (>= 0.9), geosapi, geonapi, geonode4R, ows4R

Suggests testthat, readxl, gsheets, googledrive, DBI, rapiclient, RMariaDB, RPostgres, RPostgreSQL, RSQLite, ncd4, thredds, EML, eml4, datapack, dataone, rgbif, ocs4R, zen4R, atom4R, d4storagehub4R, rmarkdown, dataverse, blastula, waldo

License MIT + file LICENSE

URL <https://github.com/r-geoflow/geoflow>

BugReports <https://github.com/r-geoflow/geoflow/issues>

LazyLoad yes

RoxygenNote 7.3.3

NeedsCompilation no

Author Emmanuel Blondel [aut, cre, cph] (ORCID: <<https://orcid.org/0000-0002-5870-5762>>),
Julien, Barde [aut] (ORCID: <<https://orcid.org/0000-0002-3519-6141>>),
Wilfried Heintz [aut] (ORCID: <<https://orcid.org/0000-0002-9244-9766>>),
Alexandre Bennici [ctb],
Sylvain Poulain [ctb],
Bastien Grasset [ctb],
Mathias Rouan [ctb],
Emilie Lerigoleur [ctb],
Yvan Le Bras [ctb],
Jeroen Ooms [ctb]

Repository CRAN

Date/Publication 2025-12-12 16:00:11 UTC

Contents

add_config_logger	5
build_hierarchical_list	5
check_packages	6
closeWorkflow	6
create_geoflow_data_from_dbi	7
create_object_identification_id	7

debugWorkflow	8
describeOGCRelation	8
enrich_text_from_entity	9
executeWorkflow	10
executeWorkflowJob	11
extract_cell_components	11
extract_kvp	12
extract_kvps	12
fetch_layer_styles_from_dbi	13
filter_sf_by_cqlfilter	13
geoflow	14
geoflowLogger	14
geoflow_action	16
geoflow_contact	20
geoflow_data	26
geoflow_data_accessor	37
geoflow_date	40
geoflow_dictionary	42
geoflow_dimension	44
geoflow_entity	46
geoflow_featuremember	58
geoflow_featuretype	60
geoflow_format	61
geoflow_handler	63
geoflow_keyword	66
geoflow_kvp	68
geoflow_process	69
geoflow_profile	71
geoflow_provenance	74
geoflow_register	75
geoflow_relation	77
geoflow_right	79
geoflow_skos_vocabulary	81
geoflow_software	84
geoflow_subject	88
geoflow_validator	91
geoflow_validator_cell	92
geoflow_validator_contacts	95
geoflow_validator_contact_Identifier	95
geoflow_validator_entities	96
geoflow_validator_entity_Creator	97
geoflow_validator_entity_Data	98
geoflow_validator_entity_Date	99
geoflow_validator_entity_Description	100
geoflow_validator_entity_Format	101
geoflow_validator_entity_Identifier	102
geoflow_validator_entity_Language	103
geoflow_validator_entity_Provenance	104

geoflow_validator_entity_Relation	105
geoflow_validator_entity_Rights	106
geoflow_validator_entity_SpatialCoverage	107
geoflow_validator_entity_Subject	108
geoflow_validator_entity_TemporalCoverage	109
geoflow_validator_entity_Title	110
geoflow_validator_entity_Type	111
geoflow_vocabulary	112
getDBTableColumnComment	113
getDBTableComment	114
get_absolute_path	114
get_config_resource_path	115
get_epsg_code	115
get_line_separator	116
get_locales_from	116
get_union_bbox	116
initWorkflow	117
initWorkflowJob	117
is_absolute_path	118
list_actions	118
list_action_options	119
list_contact_handlers	119
list_contact_handler_options	120
list_data_accessors	120
list_dictionary_handlers	121
list_dictionary_handler_options	121
list_entity_handlers	122
list_entity_handler_options	123
list_registers	123
list_software	124
list_software_parameters	124
list_software_properties	125
list_vocabularies	125
loadMetadataHandler	126
load_workflow_environment	126
posix_to_str	127
precompute_relationships	127
register_actions	128
register_contact_handlers	128
register_data_accessors	129
register_dictionary_handlers	129
register_entity_handlers	130
register_registers	130
register_software	131
register_vocabularies	131
sanitize_date	132
sanitize_str	132
set_i18n	133

<code>add_config_logger</code>	5
--------------------------------	---

check_packages	<i>check_packages</i>
----------------	-----------------------

Description

check_packages checks availability of a list of R packages in R. This function is essentially used internally by **geoflow** in association to geoflow_software and geoflow_action that would need specific packages to be imported in R.

Usage

```
check_packages(pkgs)
```

Arguments

pkgs	a vector of package names
------	---------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

closeWorkflow	<i>closeWorkflow</i>
---------------	----------------------

Description

closeWorkflow allows to close a workflow

Usage

```
closeWorkflow(config)
```

Arguments

config	a configuration object as read by closeWorkflow
--------	---

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

`create_geoflow_data_from_dbi`
create_geoflow_data_from_dbi

Description

`create_geoflow_data_from_dbi`

Usage

`create_geoflow_data_from_dbi(dbi, schema, table)`

Arguments

<code>dbi</code>	a dbi connection
<code>schema</code>	schema
<code>table</code>	table

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

`create_object_identification_id`
create_object_identification_id

Description

`create_object_identification_id`

Usage

`create_object_identification_id(prefix, str)`

Arguments

<code>prefix</code>	a character string
<code>str</code>	a character string

Value

a digested character string

debugWorkflow	<i>debugWorkflow</i>
---------------	----------------------

Description

debugWorkflow allows to initiate a workflow job for developers to work / debug on workflow actions.

Usage

```
debugWorkflow(file, dir, entityIndex, copyData, runSoftwareActions, runLocalActions)
```

Arguments

file	configuration file
dir	directory where to debug/execute the workflow
entityIndex	index of the entity within the list of loaded entities. Default is 1
copyData	whether data should be downloaded/copied to job data directory. Default is TRUE.
runSoftwareActions	whether software actions should be run. Default is TRUE.
runLocalActions	whether entity data local actions (if any) should be run. Default is TRUE

Value

a named list including the workflow config, the selected entity, the eventual options associated to the entity, and the output entity dir path of the selected entity.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

describeOGCRelation	<i>describeOGCRelation</i>
---------------------	----------------------------

Description

describeOGCRelation

Usage

```
describeOGCRelation(entity, data_object, service, download, format,
  handle_category, handle_ogc_service_description, handle_format)
```


Arguments

entity	the entity considered
data_object	data object
service	service acronym
download	whether the relation should be a download one or not
format	format
handle_category	append the relation category
handle_ogc_service_description	append the OGC service description
handle_format	append the download format

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

enrich_text_from_entity

enrich_text_from_entity

Description

enrich_text_from_entity will attempt to enrich an entity text property from other entity meta-data, depending on text variables handled by a pattern in the form

- If the entity property is a text, only the name of the property name is required.
- If the entity property is a list, then 2 subcases can be distinguished:

If it is a named list (such as entity descriptions), the text variable will be compound by the entity property name and the element_property name, in the form

If it is a unnamed list (such as list of keywords, list of relations, etc), the text variable will handle four elements: property (entity property name to look at), a key value pair to use for search within the list, an element_property for which the value should be picked up to enrich the text. The variable will be in the form

Usage

```
enrich_text_from_entity(str, entity)
```

Arguments

str	a text to be enriched
entity	an object of class geoflow_entity

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

executeWorkflow	<i>executeWorkflow</i>
-----------------	------------------------

Description

executeWorkflow allows to execute a workflow

Usage

```
executeWorkflow(file, dir, queue,  
               on_initWorkflowJob, on_initWorkflow, on_closeWorkflow,  
               monitor, session)
```

Arguments

file	a JSON geoflow configuration file
dir	a directory where to execute the workflow
queue	an ipc queue to use geoflow in geoflow-shiny
on_initWorkflowJob	a function to trigger once initWorkflowJob is executed
on_initWorkflow	a function to trigger once initWorkflow is executed
on_closeWorkflow	a function to trigger once closeWorkflow is executed
monitor	a monitor function to increase progress bar
session	a shiny session object (optional) to run geoflow in a shiny context

Value

the path of the job directory

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

executeWorkflowJob	<i>executeWorkflowJob</i>
--------------------	---------------------------

Description

executeWorkflowJob allows to execute a workflow job

Usage

```
executeWorkflowJob(config, jobdir, queue, monitor)
```

Arguments

config	a configuration object as read by initWorkflow
jobdir	the Job directory. Optional, by default inherited with the configuration.
queue	an ipc queue to use geoflow in geoflow-shiny
monitor	a monitor function to increase progress bar

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

extract_cell_components	<i>extract_cell_components</i>
-------------------------	--------------------------------

Description

extract_cell_components extracts the components of a cell when using tabular data content handlers (for entity and contact).

Usage

```
extract_cell_components(str)
```

Arguments

str	a string as object of class character
-----	---------------------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

`extract_kvp`*extract_kvp*

Description

`extract_kvp` parses a string into a key value pair represented by a `geoflow_kvp` object.

Usage

```
extract_kvp(str)
```

Arguments

<code>str</code>	a string as object of class character
------------------	---------------------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

`extract_kvps`*extract_kvps*

Description

`extract_kvp` parses a string into a key value pair represented by a `geoflow_kvp` object.

Usage

```
extract_kvps(strs, collapse)
```

Arguments

<code>strs</code>	a string as object of class character
<code>collapse</code>	collapse by. Default is NULL

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
fetch_layer_styles_from_dbi  
    fetch_layer_styles_from_dbi
```

Description

fetch_layer_styles_from_dbi

Usage

```
fetch_layer_styles_from_dbi(entity, dbi, schema, table)
```

Arguments

entity	a geoflow_entity to be used and enriched
dbi	a dbi connection
schema	schema
table	table

Value

the entity, enriched with layer styles

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
filter_sf_by_cqlfilter  
    filter_sf_by_cqlfilter
```

Description

filter_sf_by_cqlfilter filters an object of class `sf` using a CQL syntax. This function is minimalistic and only basic CQL filters are supported.

Usage

```
filter_sf_by_cqlfilter(sfdata, cqlfilter)
```

Arguments

sfdata	object of class <code>sf</code>
cqlfilter	object of class character representing a CQL filter

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

geoflow

Tools to Orchestrate and Run (Meta)Data Management Workflows

Description

Provides an engine to manage Spatial Data Infrastructures (SDI) and to orchestrate, run and automate geospatial (meta)data workflows in compliance with international standards (ISO, OGC, INSPIRE) with a range of actions such as data upload in spatial databases, publication in GeoServer and metadata publication in GeoNetwork. It includes actions to manage domain-specific resources such as ecological metadata (EML) and its publication on tools such as Metacat. It also allows to publish data on cloud data infrastructures such as Zenodo or Dataverse. Through a pivot metadata model, geoflow allows to manage a unique source dataset metadata while offering a way to target various repositories for their publication. The execution of several actions will allow to cross-share (meta)data resources in each action performed, offering a way to bind resources between each other (eg. reference Zenodo DOIS in Geonetwork/Geoserver metadata, reference Geonetwork/Geoserver links in Zenodo or EML metadata). The use of standardized configuration files allows fully reproducible workflows, in compliance with FAIR (Findable, Accessible, Interoperable, Reusable) principles.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

See Also

Useful links:

- <https://github.com/r-geoflow/geoflow>
- Report bugs at <https://github.com/r-geoflow/geoflow/issues>

geoflowLogger

geoflowLogger

Description

geoflowLogger

geoflowLogger

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling a simple logger

Public fields

verbose verbose

debug debug

Methods**Public methods:**

- [geoflowLogger\\$separator\(\)](#)
- [geoflowLogger\\$INFO\(\)](#)
- [geoflowLogger\\$WARN\(\)](#)
- [geoflowLogger\\$ERROR\(\)](#)
- [geoflowLogger\\$DEBUG\(\)](#)
- [geoflowLogger\\$new\(\)](#)
- [geoflowLogger\\$clone\(\)](#)

Method [separator\(\)](#): Util to separate chunks of logs of different natures

Usage:

`geoflowLogger$separator(char)`

Arguments:

char A character to be used, eg '='

Method [INFO\(\)](#): Prints an INFO logger message

Usage:

`geoflowLogger$INFO(txt, ...)`

Arguments:

txt logger message

... values to be passed into txt. See [sprintf](#)

Method [WARN\(\)](#): Prints an WARN logger message

Usage:

`geoflowLogger$WARN(txt, ...)`

Arguments:

txt logger message

... values to be passed into txt. See [sprintf](#)

Method [ERROR\(\)](#): Prints an ERROR logger message

Usage:

`geoflowLogger$ERROR(txt, ...)`

Arguments:

txt logger message
 ... values to be passed into txt. See [sprintf](#)

Method `DEBUG()`: Prints an DEBUG logger message

Usage:
`geoflowLogger$DEBUG(txt, ...)`

Arguments:
 txt logger message
 ... values to be passed into txt. See [sprintf](#)

Method `new()`: Initializes an object of class [geoflowLogger](#)

Usage:
`geoflowLogger$new(verbose = TRUE, debug = FALSE)`

Arguments:
 verbose TRUE if the logger is enabled, FALSE otherwise. Default is TRUE
 debug TRUE if the debugger is enabled, FALSE otherwise. Default is FALSE

Method `clone()`: The objects of this class are cloneable with this method.

Usage:
`geoflowLogger$clone(deep = FALSE)`

Arguments:
 deep Whether to make a deep clone.

Note

Logger class used internally by geoflow

geoflow_action	<i>Geoflow action class</i>
----------------	-----------------------------

Description

This class models an action to be executed by geoflow

Format

[R6Class](#) object.

Details

geoflow_action

Value

Object of [R6Class](#) for modelling an action

Super class

`geoflow::geoflowLogger -> geoflow_action`

Public fields

id action ID
 enabled enabled
 funders funders
 authors authors
 maintainer maintainer
 scope action scope
 types types of action
 def the action definition
 target the action target
 target_dir the action target directory
 packages list of packages required to perform the action
 pid_generator a name referencing the PID generator (if existing)
 pid_types types of PIDs to generate
 generic_uploader whether the action is a generic uploader or not.
 fun a function for the action
 script a script for the action
 options action options
 available_options a list of available options for the actions
 status status
 notes notes

Methods**Public methods:**

- `geoflow_action$new()`
- `geoflow_action$fromYAML()`
- `geoflow_action$checkPackages()`
- `geoflow_action$run()`
- `geoflow_action$getOption()`
- `geoflow_action$isPIDGenerator()`
- `geoflow_action$exportPIDs()`
- `geoflow_action$isGenericUploader()`
- `geoflow_action$clone()`

Method `new()`: Initialize a `geoflow_action`

Usage:

```

geoflow_action$new(
  yaml = NULL,
  id = NULL,
  enabled = TRUE,
  funders = list(),
  authors = list(),
  maintainer = NULL,
  scope = "global",
  types = list(),
  def = "",
  target = NA,
  target_dir = NA,
  packages = list(),
  pid_generator = NULL,
  pid_types = list(),
  generic_uploader = FALSE,
  fun = NULL,
  script = NULL,
  options = list(),
  available_options = list(),
  status = "stable",
  notes = ""
)

```

Arguments:

yaml a yaml file
 id action id
 enabled enabled
 funders funders
 authors authors
 maintainer maintainer
 scope action scope "global" or "local"
 types action types
 def action definition
 target the action target, e.g. "entity"
 target_dir the action target directory
 packages list of packages required to perform the action
 pid_generator a name referencing the PID generator (if existing)
 pid_types types of PIDs to generate by the action
 generic_uploader whether the action is a generic uploader or not.
 fun action function
 script action script
 options action options
 available_options available options for the action
 status status of the action (experimental, stable, deprecated, superseded)
 notes notes

Method fromYAML(): Reads action properties from YAML file

Usage:

```
geoflow_action$fromYAML(file)
```

Arguments:

file file

Method checkPackages(): Check that all packages required for the action are available, if yes, import them in the R session, and return a data.frame giving the packages names and version. If one or more packages are unavailable, an error is thrown and user informed of the missing packages.

Usage:

```
geoflow_action$checkPackages()
```

Method run(): Runs the action

Usage:

```
geoflow_action$run(entity, config)
```

Arguments:

entity entity

config config

Method getOption(): Get action option value

Usage:

```
geoflow_action$getOption(option)
```

Arguments:

option option id

Returns: the option value, either specified through a workflow, or the default value

Method isPIDGenerator(): Indicates if the action is PID generator

Usage:

```
geoflow_action$isPIDGenerator()
```

Returns: TRUE if the action is a PID generator, FALSE otherwise

Method exportPIDs(): Exports PIDs for the action. This function will export the PIDs in several ways. First, a simple CSV file including the list of PIDs for each entity, and associated status (eg. draft/release) for the given PID resource. In addition, for each metadata entities file, an equivalent metadata table will be produced as CSV to handle entities enriched with the PID (added in the "Identifier" column), ready for use as workflow entities input. In addition, a new configuration file will be produced with name "<pid_generator>_geoflow_config_for_publication.json" turned as ready for publishing resources with PIDs (eg. publishing deposits in Zenodo).

Usage:

```
geoflow_action$exportPIDs(config, entities)
```

Arguments:

config a **geoflow** configuration

entities one or more objects of class [geoflow_entity](#)

Method `isGenericUploader()`: Indicates if the action is a generic uploader

Usage:

```
geoflow_action$isGenericUploader()
```

Returns: TRUE if the action is a generic uploader, FALSE otherwise

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_action$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
action <- geoflow_action$new(
  id = "some-id",
  scope = "global",
  types = list("some purpose1", "some purpose2"),
  target = "entity",
  target_dir = "data",
  def = "some definition",
  packages = list(),
  pid_generator = NULL,
  generic_uploader = FALSE,
  fun = function(action, entity, config){},
  available_options = list(
    option_name = list(def = "option description", default = FALSE)
  ),
  options = list(option_name = TRUE)
)

## End(Not run)
```

geoflow_contact

Geoflow contact class

Description

This class models a contact to be executed by geoflow

Format

[R6Class](#) object.

Details

`geoflow_contact`

Value

Object of [R6Class](#) for modelling a contact

Public fields

`id` contact identifier
`firstName` contact first name
`lastName` contact lastname
`organizationName` contact organization
`positionName` contact position
`role` contact role
`voice` contact phone number
`facsimile` contact facsimile
`email` contact email
`websiteUrl` contact website URL
`websiteName` contact website name
`postalAddress` contact postal address
`postalCode` contact postal code
`city` contact city
`country` contact country
`identifiers` contact identifiers

Methods**Public methods:**

- [geoflow_contact\\$new\(\)](#)
- [geoflow_contact\\$setShinyEditorMode\(\)](#)
- [geoflow_contact\\$getShinyEditorMode\(\)](#)
- [geoflow_contact\\$getAllowedKeyValuesFor\(\)](#)
- [geoflow_contact\\$setIdentifier\(\)](#)
- [geoflow_contact\\$setId\(\)](#)
- [geoflow_contact\\$setFirstName\(\)](#)
- [geoflow_contact\\$setLastName\(\)](#)
- [geoflow_contact\\$setOrganizationName\(\)](#)

- `geoflow_contact$setPositionName()`
- `geoflow_contact$setRole()`
- `geoflow_contact$setVoice()`
- `geoflow_contact$setFacsimile()`
- `geoflow_contact$setEmail()`
- `geoflow_contact$setWebsiteUrl()`
- `geoflow_contact$setWebsiteName()`
- `geoflow_contact$setPostalAddress()`
- `geoflow_contact$setPostalCode()`
- `geoflow_contact$setCity()`
- `geoflow_contact$setCountry()`
- `geoflow_contact$asDataFrame()`
- `geoflow_contact$clone()`

Method `new()`: Initializes a `geoflow_contact` object

Usage:

```
geoflow_contact$new()
```

Method `setShinyEditorMode()`: Set mode for geoflow-shiny

Usage:

```
geoflow_contact$setShinyEditorMode(mode = c("creation", "edition"))
```

Arguments:

mode mode

Method `getShinyEditorMode()`: Get mode for geoflow-shiny

Usage:

```
geoflow_contact$getShinyEditorMode()
```

Returns: the shiny editor mode

Method `getAllowedKeyValuesFor()`: Retrieves keys allowed for a given tabular field name.
eg. "Identifier"

Usage:

```
geoflow_contact$getAllowedKeyValuesFor(field)
```

Arguments:

field field name

Returns: the list of valid keys for the field considered

Method `setIdentifier()`: Sets an identifier by means of key

Usage:

```
geoflow_contact$setIdentifier(key = "id", id)
```

Arguments:

key an identifier key. Default is "id"

id the identifier

Method setId(): Sets an "id" identifier

Usage:

geoflow_contact\$setId(id)

Arguments:

id the identifier

Method setFirstName(): Sets contact first name

Usage:

geoflow_contact\$setFirstName(firstName)

Arguments:

firstName contact first name

Method setLastName(): Sets contact last name

Usage:

geoflow_contact\$setLastName(lastName)

Arguments:

lastName contact last name

Method setOrganizationName(): Sets contact organization name

Usage:

geoflow_contact\$setOrganizationName(organizationName)

Arguments:

organizationName contact organization name

Method setPositionName(): Sets contact position name

Usage:

geoflow_contact\$setPositionName(positionName)

Arguments:

positionName contact position name

Method setRole(): Sets contact role

Usage:

geoflow_contact\$setRole(role)

Arguments:

role the contact role

Method setVoice(): Sets contact voice (phone number)

Usage:

geoflow_contact\$setVoice(voice)

Arguments:

voice contact voice (phone number)

Method setFacsimile(): Sets contact facsimile

Usage:

geoflow_contact\$setFacsimile(facsimile)

Arguments:

facsimile contact facsimile

Method setEmail(): Sets contact email

Usage:

geoflow_contact\$setEmail(email)

Arguments:

email contact email

Method setWebsiteUrl(): Sets contact website URL

Usage:

geoflow_contact\$setWebsiteUrl(websiteUrl)

Arguments:

websiteUrl contact website URL

Method setWebsiteName(): Sets contact website name

Usage:

geoflow_contact\$setWebsiteName(websiteName)

Arguments:

websiteName contact website name

Method setPostalAddress(): Sets the contact postal address

Usage:

geoflow_contact\$setPostalAddress(postalAddress)

Arguments:

postalAddress contact postal address

Method setPostalCode(): Sets the contact postal code

Usage:

geoflow_contact\$setPostalCode(postalCode)

Arguments:

postalCode contact postalCode

Method setCity(): Sets the contact city

Usage:

geoflow_contact\$setCity(city)

Arguments:

city contact city

Method setCountry(): Sets the contact country

Usage:

```
geoflow_contact$setCountry(country)
```

Arguments:

country contact country

Method asDataFrame(): Methods to export the [geoflow_contact](#) as data.frame using key-based syntax.

Usage:

```
geoflow_contact$asDataFrame(line_separator = NULL)
```

Arguments:

line_separator a line separator. By default, the default line separator will be used.

Returns: an object of class data.frame giving the entities using key-based syntax

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_contact$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
contact <- geoflow_contact$new()
contact$setId("john.doe@nowhere.org")
contact$setFirstName("John")
contact$setLastName("Doe")
contact$setOrganizationName("Nowhere")
contact$setPositionName("Wizard")
contact$setRole("Manager")
contact$setVoice("+9999000000000")
contact$setFacsimile("+9999000000001")
contact$setEmail("john.doe@nowhere.org")
contact$setWebsiteUrl("www.nowhere.org")
contact$setWebsiteName("Nowhere Inc.")
contact$setPostalAddress("Nowhere street")
contact$setPostalCode("Nowhere code")
contact$setCity("Nowhere city")
contact$setCountry("Nowhere country")

## End(Not run)
```

geoflow_data

Geoflow data class

Description

This class models a data object

Format

[R6Class](#) object.

Details

geoflow_data

Value

Object of [R6Class](#) for modelling a data object

Public fields

dir an object of class character giving a data directory
data list of object of class [geoflow_data](#) in case we point to a data directory
access accessor key for accessing sources. Default is 'default'
source source
sourceFid sourceFid
sourceSql sourceSql
sourceType source type
sourceZip create a zip for the sources (DEPRECATED with #344)
sourceZipOnly create a zip only for the sources, remove source files (DEPRECATED with #344)
sql sql
upload upload
uploadSource upload source name
uploadType upload type
cqlfilter CQL filter for filtering data
workspaces workspaces
store store
layername layer name
layertitle layer title
layerdesc layer description
layeruri layer URI layeruri layer URI

styles styles
 styleUpload upload styles
 dimensions dimensions
 cloud_path a relative path in a cloud storage (e.g., OCS)
 spatialRepresentationType spatial representation type eg. "vector", "grid"
 ogc_dimensions OGC dimensions
 features features
 parameters parameters
 geometryField geometry field
 geometryType geometry type
 featureType feature type name
 featureTypeObj feature type object
 attributes attributes
 variables variables
 coverages coverages
 envelopeCompositionType envelope composition type (for coverages)
 selectedResolution selected resolution (for coverages)
 selectedResolutionIndex selected resolution index (for coverages)
 bands list of bands
 actions local actions
 run whether to run local actions

Methods

Public methods:

- `geoflow_data$new()`
- `geoflow_data$getDir()`
- `geoflow_data$getData()`
- `geoflow_data$getAllowedSourceValues()`
- `geoflow_data$setAccess()`
- `geoflow_data$getAllowedSourceTypes()`
- `geoflow_data$setSourceType()`
- `geoflow_data$getAllowedGeomPossibleNames()`
- `geoflow_data$getAllowedXPossibleNames()`
- `geoflow_data$getAllowedYPossibleNames()`
- `geoflow_data$addSource()`
- `geoflow_data$setSource()`
- `geoflow_data$setSourceFid()`
- `geoflow_data$setSourceSql()`
- `geoflow_data$setSourceZip()`

- `geoflow_data$setSourceZipOnly()`
- `geoflow_data$setUploadSource()`
- `geoflow_data$getAllowedUploadTypes()`
- `geoflow_data$setUploadType()`
- `geoflow_data$setUpload()`
- `geoflow_data$setStyleUpload()`
- `geoflow_data$setSql()`
- `geoflow_data$setCqlFilter()`
- `geoflow_data$setWorkspace()`
- `geoflow_data$setStore()`
- `geoflow_data$setDatastore()`
- `geoflow_data$setLayername()`
- `geoflow_data$setLayertitle()`
- `geoflow_data$setLayerdesc()`
- `geoflow_data$setLayeruri()`
- `geoflow_data$addStyle()`
- `geoflow_data$addDimension()`
- `geoflow_data$setCloudPath()`
- `geoflow_data$getAllowedSpatialRepresentationTypes()`
- `geoflow_data$setSpatialRepresentationType()`
- `geoflow_data$setOgcDimensions()`
- `geoflow_data$setFeatures()`
- `geoflow_data$setParameter()`
- `geoflow_data$setGeometryField()`
- `geoflow_data$setGeometryType()`
- `geoflow_data$setFeatureType()`
- `geoflow_data$setFeatureTypeObj()`
- `geoflow_data$setAttributes()`
- `geoflow_data$setVariables()`
- `geoflow_data$setCoverages()`
- `geoflow_data$getAllowedEnvelopeCompositionTypes()`
- `geoflow_data$setEnvelopeCompositionType()`
- `geoflow_data$setSelectedResolution()`
- `geoflow_data$setSelectedResolutionIndex()`
- `geoflow_data$setBand()`
- `geoflow_data$addAction()`
- `geoflow_data$checkSoftwareProperties()`
- `geoflow_data$clone()`

Method `new()`: Initializes an object of class `geoflow_data`

Usage:

```
geoflow_data$new(str = NULL, config = NULL)
```

Arguments:

`str` character string to initialize from, using key-based syntax
`config` a geoflow config, if available and needed

Method `getDir()`: Get data directory where datasets are scanned to build `geoflow_data` objects

Usage:

```
geoflow_data$getDir()
```

Returns: an object of class `character`

Method `getData()`: Get a list of `geoflow_data` objects built from a directory

Usage:

```
geoflow_data$getData()
```

Returns: a list of objects of class `geoflow_data`

Method `getAllowedSourceValues()`: Get allowed source values

Usage:

```
geoflow_data$getAllowedSourceValues()
```

Returns: a vector of class `character`

Method `setAccess()`: Set accessor id. See `list_data_accessors()` for available accessors

Usage:

```
geoflow_data$setAccess(access)
```

Arguments:

`access` a data data accessor id

Method `getAllowedSourceTypes()`: Get allowed source types

Usage:

```
geoflow_data$getAllowedSourceTypes()
```

Returns: a vector of class `character`

Method `setSourceType()`: Set the source type. The source type is a simplification of the data mime type and is used to identify the type of source set for the data object. By default it is assumed to be "other" (undefined). The source types currently allowed in geoflow can be listed with `$getAllowedSourceTypes()`.

Usage:

```
geoflow_data$setSourceType(sourceType)
```

Arguments:

`sourceType` source type

Method `getAllowedGeomPossibleNames()`: Get allowed Geometry possible names for coercing data to `sf` objects

Usage:

```
geoflow_data$getAllowedGeomPossibleNames()
```

Arguments:

list of geom possible names

Method `getAllowedXPossibleNames()`: Get allowed X possible names for coercing data to **sf** objects

Usage:

```
geoflow_data$getAllowedXPossibleNames()
```

Arguments:

list of X possible names

Method `getAllowedYPossibleNames()`: Get allowed Y possible names for coercing data to **sf** objects

Usage:

```
geoflow_data$getAllowedYPossibleNames()
```

Arguments:

list of Y possible names

Method `addSource()`: Add source, object of class "character" (single source)

Usage:

```
geoflow_data$addSource(source)
```

Arguments:

source source

Method `setSource()`: Set source, object of class "character" (single source), or list. For spatial source, a single source will be used, while for sources of type 'other' (eg PDF files), multiple sources can be specified

Usage:

```
geoflow_data$setSource(source)
```

Arguments:

source source

Method `setSourceFid()`: Set source FID, object of class "character" (single source FID), or list.

Usage:

```
geoflow_data$setSourceFid(sourceFid)
```

Arguments:

sourceFid sourceFid

Method `setSourceSql()`: This is a convenience method for users that want to specify directly a SQL source. This method is called internally when a source SQL file has been set using `setSource`

Usage:

```
geoflow_data$setSourceSql(sourceSql)
```

Arguments:

sourceSql a source SQL query

Method setSourceZip(): Sets whether a zipped version of the data file(s) should be created from source files. Default is FALSE

Usage:

```
geoflow_data$setSourceZip(sourceZip)
```

Arguments:

sourceZip zip sources, object of class logical

Method setSourceZipOnly(): Sets whether a zipped version of the data file(s) only should be created from source files. Default is FALSE

Usage:

```
geoflow_data$setSourceZipOnly(sourceZipOnly)
```

Arguments:

sourceZipOnly zip sources only, object of class logical

Method setUploadSource(): Set the source to upload in output software, alternative to the source. If leave empty, the source will be used as uploadSource. A typical use case is when we want to get a CSV source to import in a database, and use the dbtable (or view/query) as upload source for publication in software like geoserver.

Usage:

```
geoflow_data$setUploadSource(uploadSource)
```

Arguments:

uploadSource upload source

Method getAllowedUploadTypes(): Get allowed upload types

Usage:

```
geoflow_data$getAllowedUploadTypes()
```

Returns: the list of allowed upload types

Method setUploadType(): The upload type is a simplification of the data mime type and is used to identify the type of data uploaded. By default it is assumed to be "other" (undefined). The upload types currently allowed in geoflow can be listed with \$getAllowedUploadTypes().

Usage:

```
geoflow_data$setUploadType(uploadType)
```

Arguments:

uploadType upload type

Method setUpload(): Set whether the source data should be uploaded to the software output declared in the geoflow configuration or not. By default it is assumed that upload should be performed (upload TRUE).

Usage:

```
geoflow_data$setUpload(upload)
```

Arguments:

upload upload

Method `setStyleUpload()`: Set whether styles in source data should be uploaded, by default TRUE

Usage:

`geoflow_data$setStyleUpload(styleUpload)`

Arguments:

styleUpload style upload

Method `setSql()`: Sets SQL for publication purpose.

Usage:

`geoflow_data$setSql(sql)`

Arguments:

sql sql

Method `setCqlFilter()`: Sets a CQL filter. In case of spatial data, once the data is read by geoflow (during initialization phase), the CQL filter will be applied to the data.

Usage:

`geoflow_data$setCqlFilter(cqlfilter)`

Arguments:

cqlfilter CQL filter

Method `setWorkspace()`: Sets a workspace name, object of class character. A workspace must target a valid software type, object of class character, to be declared as first argument of this function, assuming the corresponding software is declared in the geoflow configuration.

Usage:

`geoflow_data$setWorkspace(software_type, workspace)`

Arguments:

software_type software type where the workspace is identifier
workspace workspace name

Method `setStore()`: Sets a data/coverage store name, object of class character. Used as target data/coverage store name for GeoServer action.

Usage:

`geoflow_data$setStore(store)`

Arguments:

store store

Method `setDatastore()`: Sets a datastore name, object of class character. Used as target datastore name for GeoServer action. DEPRECATED, use `setStore`

Usage:

`geoflow_data$setDatastore(datastore)`

Arguments:

datastore datastore

Method setLayername(): Sets a layername, object of class character. Used as target layer name for Geoserver action.

Usage:

```
geoflow_data$setLayername(layername)
```

Arguments:

layername layername

Method setLayertitle(): Sets a layer title, object of class character. If available, used as target layer title in SDI-related action.

Usage:

```
geoflow_data$setLayertitle(layertitle)
```

Arguments:

layertitle layertitle

Method setLayerdesc(): Sets a layer description, object of class character. If available, used as target layer description/abstract in SDI-related actions.

Usage:

```
geoflow_data$setLayerdesc(layerdesc)
```

Arguments:

layerdesc layerdesc

Method setLayeruri(): Sets a layer URI, object of class character. If available, used as annotating URI for layer metadata (eg. in ISO 19115 action).

Usage:

```
geoflow_data$setLayeruri(layeruri)
```

Arguments:

layeruri layeruri

Method addStyle(): Adds a style name, object of class character. Used as layer style name(s) for GeoServer action.

Usage:

```
geoflow_data$addStyle(style)
```

Arguments:

style style

Method addDimension(): Adds a dimension

Usage:

```
geoflow_data$addDimension(name, dimension)
```

Arguments:

name dimension name

dimension object of class `geoflow_dimension`

Method `setCloudPath()`: Set cloud path

Usage:

```
geoflow_data$setCloudPath(cloud_path)
```

Arguments:

cloud_path cloud path

Method `getAllowedSpatialRepresentationTypes()`: Get allowed spatial representation types, typically "vector" and "grid"

Usage:

```
geoflow_data$getAllowedSpatialRepresentationTypes()
```

Returns: an object of class character

Method `setSpatialRepresentationType()`: Set spatial representation type for the data considered

Usage:

```
geoflow_data$setSpatialRepresentationType(spatialRepresentationType)
```

Arguments:

spatialRepresentationType spatial representation type

Method `setOgcDimensions()`: Set OGC dimensions

Usage:

```
geoflow_data$setOgcDimensions(name, values)
```

Arguments:

name dimension name

values dimension values

Method `setFeatures()`: Set data features

Usage:

```
geoflow_data$setFeatures(features)
```

Arguments:

features features

Method `setParameter()`: Set virtual parameter definition for setting virtual SQL view parametrized layers in Geoserver, when uploadType is set to dbquery. The arguments here follow the definition of virtual parameters in GeoServer, ie a name (alias), the corresponding field name in the DBMS, a regular expression for validation of parameter values (required to prevent SQL injection risks), and a default value.

Usage:

```
geoflow_data$setParameter(name, fieldname, regexp, defaultvalue)
```

Arguments:

name name

fieldname fieldname
regexp regexp
defaultvalue default value

Method setGeometryField(): Sets the name of the geometry field in the target GeoServer SQL view parametrized layer

Usage:

```
geoflow_data$setGeometryField(geometryField)
```

Arguments:

geometryField geometry field

Method setGeometryType(): Sets the name of the geometry field in the target GeoServer SQL view parametrized layer

Usage:

```
geoflow_data$setGeometryType(geometryType)
```

Arguments:

geometryType geometry type

Method setFeatureType(): Sets a feature type (ID) to link data with a dictionary

Usage:

```
geoflow_data$setFeatureType(featureType)
```

Arguments:

featureType feature type name

Method setFeatureTypeObj(): Sets a feature type object

Usage:

```
geoflow_data$setFeatureTypeObj(featureTypeObj)
```

Arguments:

featureTypeObj feature type object of class geoflow_featuretype

Method setAttributes(): Set attributes, as simple way to describe attributes without binding to a proper [geoflow_dictionary](#).

Usage:

```
geoflow_data$setAttributes(attributes)
```

Arguments:

attributes attributes

Method setVariables(): Set variables, as simple way to describe variables without binding to a proper [geoflow_dictionary](#).

Usage:

```
geoflow_data$setVariables(variables)
```

Arguments:

variables variables

Method setCoverages(): Set coverages

Usage:

geoflow_data\$setCoverages(coverages)

Arguments:

coverages coverages

Method getAllowedEnvelopeCompositionTypes(): Get allowed envelope composition types

Usage:

geoflow_data\$getAllowedEnvelopeCompositionTypes()

Returns: an object of class character

Method setEnvelopeCompositionType(): Set envelope composition type

Usage:

geoflow_data\$setEnvelopeCompositionType(envelopeCompositionType)

Arguments:

envelopeCompositionType envelope composition type, either 'UNION' or 'INTERSECTION'

Method setSelectedResolution(): Set selected resolution

Usage:

geoflow_data\$setSelectedResolution(selectedResolution)

Arguments:

selectedResolution selected resolution

Method setSelectedResolutionIndex(): Set selected resolution index

Usage:

geoflow_data\$setSelectedResolutionIndex(selectedResolutionIndex)

Arguments:

selectedResolutionIndex selected resolution index

Method setBand(): Set band

Usage:

geoflow_data\$setBand(name, index)

Arguments:

name band name

index band index

Method addAction(): Adds a local action

Usage:

geoflow_data\$addAction(action)

Arguments:

action object of class [geoflow_action](#)

Method `checkSoftwareProperties()`: A function triggered when loading a data object to check eventual software dependent properties, to make sure the corresponding software are declared in the config.

Usage:

```
geoflow_data$checkSoftwareProperties(config)
```

Arguments:

config geoflow config object

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_data$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_data_accessor *Geoflow data accessor class*

Description

This class models a data accessor to be used by geoflow

Format

[R6Class](#) object.

Details

geoflow_data_accessor

Value

Object of [R6Class](#) for modelling a data accessor

Super class

[geoflow::geoflowLogger](#) -> geoflow_data_accessor

Public fields

id accessor ID
 software_type accessor software type
 definition accessor definition
 packages list of packages required for the accessor
 download a download function handler
 list a function handler to list resources in case of a data directory

Methods**Public methods:**

- `geoflow_data_accessor$new()`
- `geoflow_data_accessor$setId()`
- `geoflow_data_accessor$setSoftwareType()`
- `geoflow_data_accessor$setPackages()`
- `geoflow_data_accessor$setDefinition()`
- `geoflow_data_accessor$setDownload()`
- `geoflow_data_accessor$setList()`
- `geoflow_data_accessor$checkPackages()`
- `geoflow_data_accessor$clone()`

Method `new()`: Initializes the data ccessor

Usage:

```
geoflow_data_accessor$new(
  id = NULL,
  software_type = NULL,
  definition,
  packages = list(),
  download,
  list = NULL
)
```

Arguments:

id accessor ID
 software_type accessor software type
 definition accessor definition
 packages list of packages required for the accessor
 download download function handler
 list list function handler

Method `setId()`: Sets accessor ID

Usage:

```
geoflow_data_accessor$setId(id)
```

Arguments:

id accessor ID to set

Method setSoftwareType(): Sets software type

Usage:

```
geoflow_data_accessor$setSoftwareType(software_type)
```

Arguments:

software_type software type

Method setPackages(): Sets list of packages required for the accessor

Usage:

```
geoflow_data_accessor$setPackages(packages)
```

Arguments:

packages a vector of package names

Method setDefinition(): Sets accessor definition

Usage:

```
geoflow_data_accessor$setDefinition(definition)
```

Arguments:

definition accessor definition

Method setDownload(): Set download handler (a function with arguments resource, file, path, unzip (TRUE/FALSE) and optional software)

Usage:

```
geoflow_data_accessor$setDownload(download)
```

Arguments:

download an object of class function

Method setList(): Set list handler (a function with no arguments)

Usage:

```
geoflow_data_accessor$setList(list)
```

Arguments:

list an object of class function

Method checkPackages(): Check that all packages required for the software are available, if yes, import them in the R session, and return a data.frame giving the packages names and version. If one or more packages are unavailable, an error is thrown and user informed of the missing packages.

Usage:

```
geoflow_data_accessor$checkPackages()
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_data_accessor$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
access <- geoflow_data_accessor$new(
  id = "some-id",
  software_type = "some-software",
  definition = "definition",
  packages = list(),
  download = function(resource, file, path, software, unzip){},
  list = function(resource, software){}
)

## End(Not run)
```

geoflow_date

Geoflow date class

Description

This class models an date

Format

[R6Class](#) object.

Details

geoflow_date

Value

Object of [R6Class](#) for modelling an date

Public fields

key date key. Default is "creation"

value date value. Default is generated with Sys.time()

Methods

Public methods:

- `geoflow_date$new()`
- `geoflow_date$setKey()`
- `geoflow_date$setValue()`
- `geoflow_date$clone()`

Method `new()`: Initializes a `geoflow_date`

Usage:

```
geoflow_date$new()
```

Method `setKey()`: Sets the date key

Usage:

```
geoflow_date$setKey(key)
```

Arguments:

key date key

Method `setValue()`: Sets the date value. The method will check validity of date value.

Usage:

```
geoflow_date$setValue(value)
```

Arguments:

value date value

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_date$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
date <- geoflow_date$new()
date$setKey("creation")
date$setValue(Sys.time())

## End(Not run)
```

geoflow_dictionary	<i>Geoflow dictionary class</i>
--------------------	---------------------------------

Description

This class models a dictionary to be executed by geoflow

Format

[R6Class](#) object.

Details

geoflow_dictionary

Value

Object of [R6Class](#) for modelling a dictionary

Public fields

source dictionary source, object of class data.frame

featuretypes list of objects of class geoflow_featuretype

registers list of objects of class geoflow_register

Methods

Public methods:

- [geoflow_dictionary\\$new\(\)](#)
- [geoflow_dictionary\\$setSource\(\)](#)
- [geoflow_dictionary\\$getFeatureTypes\(\)](#)
- [geoflow_dictionary\\$getFeatureTypeById\(\)](#)
- [geoflow_dictionary\\$addFeatureType\(\)](#)
- [geoflow_dictionary\\$getRegisters\(\)](#)
- [geoflow_dictionary\\$getRegisterById\(\)](#)
- [geoflow_dictionary\\$addRegister\(\)](#)
- [geoflow_dictionary\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_dictionary](#) object

Usage:

```
geoflow_dictionary$new()
```

Method [setSource\(\)](#): Sets dictionary source

Usage:

```
geoflow_dictionary$setSource(source)
```

Arguments:

source object of class `data.frame`

Method `getFeatureTypes()`: Get the list of [geoflow_featuretype](#) defined in the dictionary

Usage:

```
geoflow_dictionary$getFeatureTypes()
```

Returns: a list of `geoflow_featuretype`

Method `getFeatureTypeById()`: Get an object of class [geoflow_featuretype](#) given an ID

Usage:

```
geoflow_dictionary$getFeatureTypeById(id)
```

Arguments:

id id

Returns: an object of class [geoflow_featuretype](#), otherwise NULL

Method `addFeatureType()`: Adds a feature type to the dictionary

Usage:

```
geoflow_dictionary$addFeatureType(ft)
```

Arguments:

ft object of class [geoflow_featuretype](#)

Method `getRegisters()`: Get the list of registers associated with the dictionary

Usage:

```
geoflow_dictionary$getRegisters()
```

Returns: a list of [geoflow_register](#) objects

Method `getRegisterById()`: Get register by ID

Usage:

```
geoflow_dictionary$getRegisterById(id)
```

Arguments:

id id

Returns: an object of class [geoflow_register](#), otherwise NULL

Method `addRegister()`: Adds a register to the dictionary

Usage:

```
geoflow_dictionary$addRegister(register)
```

Arguments:

register object of class [geoflow_register](#)

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_dictionary$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_dimension	<i>Geoflow dimension class</i>
-------------------	--------------------------------

Description

This class models a dimension

Format

[R6Class](#) object.

Details

geoflow_dimension

Value

Object of [R6Class](#) for modelling a dimension

Public fields

longName dimension longName
resolution dimension resolution
size dimension size
values dimension values
minValue dimension min value
maxValue dimension max value

Methods**Public methods:**

- [geoflow_dimension\\$new\(\)](#)
- [geoflow_dimension\\$setLongName\(\)](#)
- [geoflow_dimension\\$setResolution\(\)](#)
- [geoflow_dimension\\$setSize\(\)](#)
- [geoflow_dimension\\$setValues\(\)](#)
- [geoflow_dimension\\$setMinValue\(\)](#)
- [geoflow_dimension\\$setMaxValue\(\)](#)
- [geoflow_dimension\\$clone\(\)](#)

Method [new\(\)](#): Initializes the [geoflow_dimension](#)

Usage:

```
geoflow_dimension$new()
```

Method setLongName(): Sets the dimension long name

Usage:

```
geoflow_dimension$setLongName(longName)
```

Arguments:

longName dimension long name

Method setResolution(): Sets the resolution

Usage:

```
geoflow_dimension$setResolution(uom, value)
```

Arguments:

uom unit of measure

value resolution value

Method setSize(): Sets the dimension size

Usage:

```
geoflow_dimension$setSize(size)
```

Arguments:

size dimension size

Method setValues(): Sets dimension values

Usage:

```
geoflow_dimension$setValues(values)
```

Arguments:

values dimension values

Method setMinValue(): Sets dimension min value

Usage:

```
geoflow_dimension$setMinValue(minValue)
```

Arguments:

minValue min value

Method setMaxValue(): Sets dimension max value

Usage:

```
geoflow_dimension$setMaxValue(maxValue)
```

Arguments:

maxValue max value

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_dimension$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
dimension <- geoflow_dimension$new()
dimension$setLongName("longname")
dimension$setResolution(uom="s", value=1)
dimension$setSize(10)
dimension$setValues(c(1,2,3))
dimension$setMinValue(1)
dimension$setMaxValue(3)

## End(Not run)
```

geoflow_entity

Geoflow entity class

Description

This class models a entity object

Format

[R6Class](#) object.

Details

geoflow_entity

Value

Object of [R6Class](#) for modelling a entity object

Public fields

identifiers entity identifiers
 dates entity dates
 language entity language
 types entity types
 titles entity titles
 descriptions entity descriptions
 subjects entity subjects
 formats entity formats

contacts entity contacts
relations entity relations
rights entity rights
spatial_extent spatial extent
spatial_bbox spatial bounding box
geo_bbox geographic bounding box (in EPSG:4326 / WGS 84)
srid entity srid
temporal_extent entity temporal extent
provenance entity provenance
data entity data
status entity status
resources entity resources
locales entity locales

Methods

Public methods:

- `geoflow_entity$new()`
- `geoflow_entity$setShinyEditorMode()`
- `geoflow_entity$getShinyEditorMode()`
- `geoflow_entity$getAllowedKeyValuesFor()`
- `geoflow_entity$addLocalesFromValues()`
- `geoflow_entity$setIdentifier()`
- `geoflow_entity$addDate()`
- `geoflow_entity$setLanguage()`
- `geoflow_entity$writeDataResource()`
- `geoflow_entity$setType()`
- `geoflow_entity$setTitle()`
- `geoflow_entity$setDescription()`
- `geoflow_entity$addSubject()`
- `geoflow_entity$addFormat()`
- `geoflow_entity$addContact()`
- `geoflow_entity$addRelation()`
- `geoflow_entity$addRight()`
- `geoflow_entity$setSpatialExtent()`
- `geoflow_entity$setSpatialBbox()`
- `geoflow_entity$setGeographicBbox()`
- `geoflow_entity$setSrid()`
- `geoflow_entity$setTemporalExtent()`
- `geoflow_entity$setProvenance()`
- `geoflow_entity$setData()`

- `geoflow_entity$addData()`
- `geoflow_entity$getEntityJobDirname()`
- `geoflow_entity$getEntityJobDirPath()`
- `geoflow_entity$prepareEntityJobDir()`
- `geoflow_entity$copyDataToJobDir()`
- `geoflow_entity$copyStylesToJobDir()`
- `geoflow_entity$enrichWithDatatypes()`
- `geoflow_entity$enrichWithData()`
- `geoflow_entity$enrichSpatialCoverageFromDB()`
- `geoflow_entity$enrichWithFeatures()`
- `geoflow_entity$enrichWithCoverages()`
- `geoflow_entity$prepareFeaturesToUpload()`
- `geoflow_entity$enrichWithIdentifiers()`
- `geoflow_entity$enrichWithRelations()`
- `geoflow_entity$enrichWithSubjects()`
- `geoflow_entity$enrichWithVocabularies()`
- `geoflow_entity$enrichWithFormats()`
- `geoflow_entity$enrichWithMetadata()`
- `geoflow_entity$getContacts()`
- `geoflow_entity$getSubjects()`
- `geoflow_entity$getRelations()`
- `geoflow_entity$setStatus()`
- `geoflow_entity$getJobResource()`
- `geoflow_entity$getJobDataResource()`
- `geoflow_entity$getJobMetadataResource()`
- `geoflow_entity$addResource()`
- `geoflow_entity$asDataFrame()`
- `geoflow_entity$clone()`

Method `new()`: Initializes an object of class `geoflow_entity`

Usage:

```
geoflow_entity$new()
```

Method `setShinyEditorMode()`: Set mode for geoflow-shiny

Usage:

```
geoflow_entity$setShinyEditorMode(mode = c("creation", "edition"))
```

Arguments:

mode mode

Method `getShinyEditorMode()`: Get mode for geoflow-shiny

Usage:

```
geoflow_entity$getShinyEditorMode()
```

Returns: the shiny editor mode

Method `getAllowedKeyValuesFor()`: Retrieves keys allowed for a given tabular field name. eg. "Identifier"

Usage:

```
geoflow_entity$getAllowedKeyValuesFor(field)
```

Arguments:

field field name

Returns: the list of valid keys for the field considered

Method `addLocalesFromValues()`: Adds locales to entity from kvp values

Usage:

```
geoflow_entity$addLocalesFromValues(values)
```

Arguments:

values values

Method `setIdentifier()`: Set an identifier given a key. Default key is "id", but others can be specified, eg "doi".

Usage:

```
geoflow_entity$setIdentifier(key = "id", id)
```

Arguments:

key identifier key. Default is "id"

id identifier value

Method `addDate()`: Adds a date

Usage:

```
geoflow_entity$addDate(dateType, date)
```

Arguments:

dateType date type, object of class character

date date, object of class Date or POSIXt

Method `setLanguage()`: Set the language used for the entity description (metadata). Default is "eng".

Usage:

```
geoflow_entity$setLanguage(language)
```

Arguments:

language language

Method `writeDataResource()`: writes a data resource. Deprecrated Note: TODO to review in line with 'writeWorkflowJobDataResource

Usage:

```
geoflow_entity$writeDataResource(obj = NULL, resourcename, type = "shp")
```

Arguments:

obj object

resourcename resource name
type type of resource

Method setType(): Set the type of description. By default a generic type (key = "generic") is defined to "dataset", and will be used as default type for actions that perform metadata production / publication.

Usage:
geoflow_entity\$setType(key = "generic", type)

Arguments:
key type key. Default is "generic"
type type value

Method setTitle(): Sets title

Usage:
geoflow_entity\$setTitle(key = "title", title)

Arguments:
key title key. Default is "title"
title title value

Method setDescription(): Sets description

Usage:
geoflow_entity\$setDescription(key, description)

Arguments:
key description key. Default is "abstract"
description description value

Method addSubject(): Adds a subject

Usage:
geoflow_entity\$addSubject(subject)

Arguments:
subject object of class [geoflow_subject](#)

Method addFormat(): Adds a format

Usage:
geoflow_entity\$addFormat(format)

Arguments:
format object of class [geoflow_format](#)

Method addContact(): Adds a contact

Usage:
geoflow_entity\$addContact(contact)

Arguments:
contact object of class [geoflow_contact](#)

Method addRelation(): Adds a relation

Usage:

```
geoflow_entity$addRelation(relation)
```

Arguments:

relation object of class [geoflow_relation](#)

Method addRight(): Adds a right

Usage:

```
geoflow_entity$addRight(right)
```

Arguments:

right object of class [geoflow_right](#)

Method setSpatialExtent(): Set spatial extent. Various ways can be used to set the spatial extent 1) with a WKT string, 2) with a bbox, object of class `matrix`, or 3) specifying a data object (from `sf`). The crs (coordinate reference system) should be specified with the crs SRID (number). The spatial extent is not necessarily a bounding box but can be one or more geometries.

Usage:

```
geoflow_entity$setSpatialExtent(wkt = NULL, bbox = NULL, data = NULL, crs = NA)
```

Arguments:

wkt a WKT string

bbox a bbox

data an object of class `sf`

crs crs

Method setSpatialBbox(): Set spatial bbox. Various ways can be used to set the spatial extent 1) with a WKT string, 2) with a bbox, object of class `matrix`, or 3) specifying a data object (from `sf`). The crs (coordinate reference system) should be specified with the crs SRID (number).

Usage:

```
geoflow_entity$setSpatialBbox(wkt = NULL, bbox = NULL, data = NULL, crs = NA)
```

Arguments:

wkt a WKT string

bbox a bbox

data an object of class `sf`

crs crs

Method setGeographicBbox(): Set geographic bbox (in EPSG:4326 / WGS 84), by converting (if needed) the spatial bbox

Usage:

```
geoflow_entity$setGeographicBbox()
```

Method setSrid(): Sets entity SRID

Usage:

```
geoflow_entity$setSrid(srid)
```

Arguments:

srid srid

Method setTemporalExtent(): Sets temporal extent. The temporal extent can be a year, date instant or interval

Usage:

geoflow_entity\$setTemporalExtent(str)

Arguments:

str object of class numeric (case of year) or character

Method setProvenance(): Sets entity provenance

Usage:

geoflow_entity\$setProvenance(provenance)

Arguments:

provenance object of class [geoflow_provenance](#)

Method setData(): Sets entity data object

Usage:

geoflow_entity\$setData(data)

Arguments:

data object of class [geoflow_data](#)

Method addData(): Adds entity data object

Usage:

geoflow_entity\$addData(data)

Arguments:

data object of class [geoflow_data](#)

Method getEntityJobDirname(): Gets entity job directory name. In case entity is identified with a DOI, the '/' (slash) will be replaced by '_' (underscore) to make sure directory is created.

Usage:

geoflow_entity\$getEntityJobDirname()

Returns: get the name of entity job directory that will be created for the entity

Method getEntityJobDirPath(): Gets entity job directory path. In the job directory, all entities subdirs will be created within a 'entities' directory.

Usage:

geoflow_entity\$getEntityJobDirPath(config, jobdir = NULL)

Arguments:

config geoflow configuration object

jobdir relative path of the job directory

Returns: the entity job directory path

Method prepareEntityJobDir(): Function called internally by **geoflow** that creates the entity directory and relevant sub-directories. The default sub-directories will include 'data' and 'meta-data'. Other sub-directories may be created depending on the actions enabled in the workflow (and if their target directory is different from 'data'/'metadata').

Usage:

```
geoflow_entity$prepareEntityJobDir(config, jobdir = NULL)
```

Arguments:

config geoflow config object
jobdir relative path of the job directory

Method copyDataToJobDir(): This function will look at data object(s) associated to the entity (previously set with setData or added with addData), and will try to (download)/copy the data source to the geoflow job directory.

Usage:

```
geoflow_entity$copyDataToJobDir(config, jobdir = NULL)
```

Arguments:

config geoflow config object
jobdir relative path of the job directory

Method copyStylesToJobDir(): This function checks for the availability of layer styles (set as entity resource) that would have been added with DBI handlers from a special DB 'layer_styles' table

Usage:

```
geoflow_entity$copyStylesToJobDir(config)
```

Arguments:

config geoflow config object

Method enrichWithDatatypes(): Function that will scan zip data files and resolve data objects sourceType and uploadType

Usage:

```
geoflow_entity$enrichWithDatatypes(config, jobdir = NULL)
```

Arguments:

config geoflow config object
jobdir relative path of the job directory

Method enrichWithData(): This function will enrich the entity data objects with data features (vector data) or coverages (grid data). This method will overwrite spatial metadata such as the bounding box (unless global option skipDynamicBbox is enabled). Note that the user spatial extent is not overwritten since it may contain finer geometries than a bounding box.

Usage:

```
geoflow_entity$enrichWithData(config, jobdir = NULL)
```

Arguments:

config geoflow config object

jobdir relative path of the job directory

Method `enrichSpatialCoverageFromDB()`: This function computes spatial coverage from DB (table, view or query) without having to deal with a full data download. It is triggered when the global option `skipDataDownload` is enabled.

Usage:

```
geoflow_entity$enrichSpatialCoverageFromDB(config)
```

Arguments:

config geoflow config object

Method `enrichWithFeatures()`: This function will enrich the entity data objects with data features (vector data). This method will overwrite spatial metadata such as the bounding box (unless global option `skipDynamicBbox` is enabled). Note that the user spatial extent is not overwritten since it may contain finer geometries than a bounding box.

Usage:

```
geoflow_entity$enrichWithFeatures(config, jobdir = NULL)
```

Arguments:

config geoflow config object

jobdir relative path of the job directory

Method `enrichWithCoverages()`: This function will enrich the entity data objects with data coverages (grid data). This method will overwrite spatial metadata such as the bounding box (unless global option `skipDynamicBbox` is enabled). Note that the user spatial extent is not overwritten since it may contain finer geometries than a bounding box.

Usage:

```
geoflow_entity$enrichWithCoverages(config, jobdir = NULL)
```

Arguments:

config geoflow config object

jobdir relative path of the job directory

Method `prepareFeaturesToUpload()`: This function will 1) check (in case of upload is requested) if the type of source and upload are both different on files formats(eg. csv,shp,gpkg) and 2) process automatically to conversion from source to upload type.

Usage:

```
geoflow_entity$prepareFeaturesToUpload(config)
```

Arguments:

config geoflow config object

Method `enrichWithIdentifiers()`: Function that will enrich entity with identifiers needed across multiple actions

Usage:

```
geoflow_entity$enrichWithIdentifiers(config)
```

Arguments:

config geoflow config object

Method `enrichWithRelations()`: This function that will enrich the entity with relations. At now this is essentially related to adding relations if a Geoserver (geosapi) publishing action is enabled. Relations added will depend on the `enrich_with_relation_*` options set in a) the geosapi action, ie. 1) add WMS auto-generated thumbnail (if option `enrich_with_relation_wms_thumbnail` is TRUE) 2) add WMS base URL relation (if option `enrich_with_relation_wms` is TRUE) 3) for vector spatial representation type: - add WFS base URL relation (if option `enrich_with_relation_wfs` is TRUE) - add WFS auto-generated links as convenience for data download links (if option `enrich_with_relation_wfs_download_links` is TRUE) 4) for grid spatial representation type: - add WCS base URL relation (if option `enrich_with_relation_wcs` is TRUE) b) in the geonapi action (for adding a CSW metadata URL) b) in the ows4R action (for adding a CSW metadata URL)

Usage:

```
geoflow_entity$enrichWithRelations(config)
```

Arguments:

config geoflow config object

Method `enrichWithSubjects()`: Enrichs the entity with subjects. If no subject specify in Subjects, automatically add keyword from dictionary to 'theme' category

Usage:

```
geoflow_entity$enrichWithSubjects(config, exclusions = c())
```

Arguments:

config geoflow config object

exclusions exclusions

Method `enrichWithVocabularies()`: Enrichs the entity with vocabularies

Usage:

```
geoflow_entity$enrichWithVocabularies(config)
```

Arguments:

config geoflow config object

Method `enrichWithFormats()`: Enrichs the entity with formats

Usage:

```
geoflow_entity$enrichWithFormats(config)
```

Arguments:

config geoflow config object

Method `enrichWithMetadata()`: Enrichs the entity properties with entity metadata from other properties.

Usage:

```
geoflow_entity$enrichWithMetadata(config)
```

Arguments:

config geoflow config object

Method `getContacts()`: Get the entity contacts

Usage:

```
geoflow_entity$getContacts(pretty = FALSE)
```

Arguments:

pretty to prettify the output as data.frame

Returns: a list of geoflow_contact or a data.frame

Method getSubjects(): Get the entity subjects

Usage:

```
geoflow_entity$getSubjects(pretty = FALSE, keywords = FALSE)
```

Arguments:

pretty to prettify the output as data.frame

keywords to add keywords to the output or not. Default is FALSE

Returns: a list of geoflow_subject or a data.frame

Method getRelations(): Get the entity relations

Usage:

```
geoflow_entity$getRelations(pretty = FALSE)
```

Arguments:

pretty to prettify the output as data.frame

Returns: a list of geoflow_relation or a data.frame

Method setStatus(): Set a simple status either "draft" or "published". This method is required to deal with systems that manage DOIs, such as Zenodo (with **zen4R**) or Dataverse (with **atom4R**) publishing actions (Used internally by **geoflow**).

Usage:

```
geoflow_entity$setStatus(system, status)
```

Arguments:

system a system name eg. "zenodo", "dataverse"

status a status for entity resource "draft" or "published"

Method getJobResource(): Get an entity job resource path

Usage:

```
geoflow_entity$getJobResource(config, resourceType, filename)
```

Arguments:

config a geoflow config object

resourceType type of resource, matching a sub-directory within the entity job directory

filename filename

Returns: the file path of the job resource

Method getJobDataResource(): Get an entity job data resource path

Usage:

```
geoflow_entity$getJobDataResource(config, filename)
```


Arguments:

config a geoflow config object

filename filename

Returns: the file path of the job data resource

Method getJobMetadataResource(): Get an entity job metadata resource path

Usage:

```
geoflow_entity$getJobMetadataResource(config, filename)
```

Arguments:

config a geoflow config object

filename filename

Returns: the file path of the job metadata resource

Method addResource(): Adds a resource to the entity

Usage:

```
geoflow_entity$addResource(id, resource)
```

Arguments:

id id of the resource

resource resource

Method asDataFrame(): Methods to export the [geoflow_entity](#) as data.frame using key-based syntax.

Usage:

```
geoflow_entity$asDataFrame(line_separator = NULL)
```

Arguments:

line_separator a line separator. By default, the default line separator will be used.

Returns: an object of class data.frame giving the entities using key-based syntax

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_entity$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_featuremember *Geoflow feature type class*

Description

This class models a feature type to be executed by geoflow

Format

[R6Class](#) object.

Details

geoflow_featuremember

Value

Object of [R6Class](#) for modelling a dictionary feature type member

Public fields

id feature member ID
type feature member type
code feature member code
name feature member name
def feature member definition
defSource feature member definition source
minOccurs feature member minOccurs
maxOccurs feature member maxOccurs
uom feature member unit of measure (uom)
registerId feature member register ID
registerScript feature member register script

Methods

Public methods:

- [geoflow_featuremember\\$new\(\)](#)
- [geoflow_featuremember\\$asDataFrame\(\)](#)
- [geoflow_featuremember\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_featuremember](#)

Usage:

```
geoflow_featuremember$new(  
  type = "attribute",  
  code = NULL,  
  name = NULL,  
  def = NULL,  
  defSource = NULL,  
  minOccurs = NULL,  
  maxOccurs = NULL,  
  uom = NULL,  
  registerId = NULL,  
  registerScript = NULL  
)
```

Arguments:

type type

code code

name name

def definition

defSource definition source. Default is NULL

minOccurs minOccurs. Default is NULL

maxOccurs maxOccurs. Default is NULL

uom unit of measure. Default is NULL

registerId ID of the register associated to the feature type. Default is NULL

registerScript source script providing the register functions. Default is NULL

Method asDataFrame(): Converts as data.frame

Usage:

```
geoflow_featuremember$asDataFrame()
```

Returns: an object of class [data.frame](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_featuremember$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_featuretype	<i>Geoflow feature type class</i>
---------------------	-----------------------------------

Description

This class models a feature type to be executed by geoflow

Format

[R6Class](#) object.

Details

geoflow_featuretype

Value

Object of [R6Class](#) for modelling a dictionary feature type

Public fields

id feature type ID

members feature type members

Methods

Public methods:

- [geoflow_featuretype\\$new\(\)](#)
- [geoflow_featuretype\\$addMember\(\)](#)
- [geoflow_featuretype\\$getMembers\(\)](#)
- [geoflow_featuretype\\$getMemberById\(\)](#)
- [geoflow_featuretype\\$asDataFrame\(\)](#)
- [geoflow_featuretype\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_featuretype](#)

Usage:

```
geoflow_featuretype$new(id = NULL)
```

Arguments:

id id

Method [addMember\(\)](#): Adds a member

Usage:

```
geoflow_featuretype$addMember(fm)
```

Arguments:

fm object of class [geoflow_featuremember](#)

Method getMembers(): Get members

Usage:

geoflow_featuretype\$getMembers()

Returns: the list of members, as objects of class [geoflow_featuremember](#)

Method getMemberById(): Get member by ID

Usage:

geoflow_featuretype\$getMemberById(id)

Arguments:

id id

Returns: an object of class [geoflow_featuremember](#), NULL otherwise

Method asDataFrame(): Converts to data frame

Usage:

geoflow_featuretype\$asDataFrame()

Returns: an object of class [data.frame](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

geoflow_featuretype\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_format	<i>Geoflow format class</i>
----------------	-----------------------------

Description

This class models a format

Format

[R6Class](#) object.

Details

geoflow_format

Value

Object of [R6Class](#) for modelling a format

Public fields

key format key
name key name
uri key URI
description key description

Methods**Public methods:**

- [geoflow_format\\$new\(\)](#)
- [geoflow_format\\$setKey\(\)](#)
- [geoflow_format\\$setName\(\)](#)
- [geoflow_format\\$setUri\(\)](#)
- [geoflow_format\\$setDescription\(\)](#)
- [geoflow_format\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_format](#)

Usage:

```
geoflow_format$new(str = NULL)
```

Arguments:

str character string to initialize object using key-based syntax

Method [setKey\(\)](#): Sets format key

Usage:

```
geoflow_format$setKey(key)
```

Arguments:

key key

Method [setName\(\)](#): Sets format name

Usage:

```
geoflow_format$setName(name)
```

Arguments:

name name

Method [setUri\(\)](#): Sets format URI

Usage:

```
geoflow_format$setUri(uri)
```

Arguments:

uri URI

Method setDescription(): Sets format description

Usage:

```
geoflow_format$setDescription(description)
```

Arguments:

description description

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_format$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
format <- geoflow_format$new()
format$setKey("distribution")
format$setName("text/csv")
format$setUri("https://www.iana.org/assignments/media-types/text/csv")
format$setDescription("CSV format")

## End(Not run)
```

geoflow_handler	<i>Geoflow handler class</i>
-----------------	------------------------------

Description

This class models a content handler. An handler is a method to handle some content (eg entity or contact). It is mainly driven by a function that takes as argument the handler considered (as self accessible object), a source which identifies the source to be handled, that can be of a different type (eg a URL, a file path) depending on the handler, and a config object, as the overall configuration created by geoflow initWorkflow function.

Format

R6Class object.

Details

geoflow_handler

Value

Object of [R6Class](#) for modelling a handler

Public fields

id handler id
 type handler type (entity,contact,dictionary)
 funders funders
 authors authors
 maintainer maintainer
 def handler definition
 packages handler packages
 fun handler function
 script handler script
 options options
 available_options available options
 status status
 notes notes

Methods**Public methods:**

- [geoflow_handler\\$new\(\)](#)
- [geoflow_handler\\$fromYAML\(\)](#)
- [geoflow_handler\\$checkPackages\(\)](#)
- [geoflow_handler\\$getOption\(\)](#)
- [geoflow_handler\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_handler](#)

Usage:

```
geoflow_handler$new(
  yaml = NULL,
  id = NULL,
  type = c("entity", "contact", "dictionary"),
  funders = list(),
  authors = list(),
  maintainer = NULL,
  def = "",
  packages = list(),
  fun = NULL,
  script = NULL,
  options = list(),
  available_options = list(),
```



```

        status = "stable",
        notes = ""
    )

```

Arguments:

- yaml a YAML file
- id id
- type type
- funders funders
- authors authors
- maintainer maintainer
- def def
- packages list of packages required for the handler
- fun the handler function having 2 arguments config and source
- script a handler script
- options action options
- available_options available options for the action
- status status (experimental/stable/deprecated/superseded)
- notes notes

Method fromYAML(): Reads handler properties from YAML file

Usage:

```
geoflow_handler$fromYAML(file)
```

Arguments:

file file

Method checkPackages(): Check that all packages required for the handler are available, if yes, import them in the R session, and return a data.frame giving the packages names and version. If one or more packages are unavailable, an error is thrown and user informed of the missing packages.

Usage:

```
geoflow_handler$checkPackages()
```

Method getOption(): Get handler option value

Usage:

```
geoflow_handler$getOption(option)
```

Arguments:

option option id

Returns: the option value, either specified through a workflow, or the default value

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_handler$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Note

This class is essentially called internally by geoflow to register default handlers for entities and contacts.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
handler <- geoflow_handler$new(
  id = "some-id",
  def = "some definition",
  packages = list(),
  fun = function(handler, source, config){},
  available_options = list()
)

## End(Not run)
```

geoflow_keyword

Geoflow keyword class

Description

This class models a keyword

Format

[R6Class](#) object.

Details

geoflow_keyword

Value

Object of [R6Class](#) for modelling a keyword

Public fields

name keyword

uri keyword uri

Methods

Public methods:

- `geoflow_keyword$new()`
- `geoflow_keyword$setName()`
- `geoflow_keyword$setUri()`
- `geoflow_keyword$clone()`

Method `new()`: Initializes a `geoflow_keyword`

Usage:

```
geoflow_keyword$new(name = NULL, uri = NULL)
```

Arguments:

name keyword name

uri keyword URI

Method `setName()`: Sets keyword

Usage:

```
geoflow_keyword$setName(name)
```

Arguments:

name keyword name

Method `setUri()`: Sets keyword URI

Usage:

```
geoflow_keyword$setUri(uri)
```

Arguments:

uri keyword URI

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_keyword$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
kwd <- geoflow_keyword$new()
kwd$setName("keyword")
kwd$setUri("http://somelink/keyword")

## End(Not run)
```

geoflow_kvp	<i>Geoflow kvp (Key Values pair) class</i>
-------------	--

Description

This class models an kvp (Key Values pair)

Format

[R6Class](#) object.

Details

geoflow_kvp

Value

Object of [R6Class](#) for modelling an kvp (Key Values pair)

Public fields

key the KVP key
values the KVP values
locale a locale definition for the KVP

Methods**Public methods:**

- [geoflow_kvp\\$new\(\)](#)
- [geoflow_kvp\\$setKey\(\)](#)
- [geoflow_kvp\\$setValues\(\)](#)
- [geoflow_kvp\\$setLocale\(\)](#)
- [geoflow_kvp\\$clone\(\)](#)

Method [new\(\)](#): Initializes a Key-Value pair (KVP)

Usage:

```
geoflow_kvp$new(key = NULL, values = NULL, locale = NULL)
```

Arguments:

key key
values values
locale locale

Method [setKey\(\)](#): Set KVP key

Usage:

```
geoflow_kvp$setKey(key)
```

Arguments:

key the key

Method setValues(): Set KVP values

Usage:

geoflow_kvp\$setValues(values)

Arguments:

values the values

Method setLocale(): Set KVP locale

Usage:

geoflow_kvp\$setLocale(locale)

Arguments:

locale locale

Method clone(): The objects of this class are cloneable with this method.

Usage:

geoflow_kvp\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

geoflow_process

Geoflow process class

Description

This class models an process

Format

[R6Class](#) object.

Details

geoflow_process

Value

Object of [R6Class](#) for modelling an process

Public fields

rationale process rationale
description process description
processors object of class list

Methods**Public methods:**

- [geoflow_process\\$new\(\)](#)
- [geoflow_process\\$setRationale\(\)](#)
- [geoflow_process\\$setDescription\(\)](#)
- [geoflow_process\\$addProcessor\(\)](#)
- [geoflow_process\\$clone\(\)](#)

Method [new\(\)](#): Initializes the [geoflow_process](#)

Usage:

`geoflow_process$new()`

Method [setRationale\(\)](#): Set process rationale

Usage:

`geoflow_process$setRationale(rationale)`

Arguments:

rationale the process rationale

Method [setDescription\(\)](#): Set process description

Usage:

`geoflow_process$setDescription(description)`

Arguments:

description Set the process description

Method [addProcessor\(\)](#): Adds processor

Usage:

`geoflow_process$addProcessor(processor)`

Arguments:

processor, object of class [geoflow_contact](#)

Method [clone\(\)](#): The objects of this class are cloneable with this method.

Usage:

`geoflow_process$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
process <- geoflow_process$new()
process$setRationale("rationale")
process$setDescription("description")
processor <- geoflow_contact$new()
process$addProcessor(processor)

## End(Not run)
```

geoflow_profile	<i>Geoflow profile class</i>
-----------------	------------------------------

Description

This class models an profile

Format

[R6Class](#) object.

Details

geoflow_profile

Value

Object of [R6Class](#) for modelling an profile

Public fields

id profile id
 name profile name
 project profile project
 organization profile organization
 logos profile logo(s)
 mode mode of execution (Default is "raw")
 options global config options

Methods

Public methods:

- `geoflow_profile$new()`
- `geoflow_profile$setId()`
- `geoflow_profile$setName()`
- `geoflow_profile$setProject()`
- `geoflow_profile$setOrganization()`
- `geoflow_profile$addLogo()`
- `geoflow_profile$setMode()`
- `geoflow_profile$setOption()`
- `geoflow_profile$clone()`

Method `new()`: Initializes an object of class `geoflow_profile`

Usage:

```
geoflow_profile$new()
```

Method `setId()`: Sets profile ID

Usage:

```
geoflow_profile$setId(id)
```

Arguments:

`id` `id`

Method `setName()`: Sets profile name

Usage:

```
geoflow_profile$setName(name)
```

Arguments:

`name` `name`

Method `setProject()`: Sets profile project

Usage:

```
geoflow_profile$setProject(project)
```

Arguments:

`project` `project`

Method `setOrganization()`: Sets profile organization

Usage:

```
geoflow_profile$setOrganization(organization)
```

Arguments:

`organization` `organization`

Method `addLogo()`: Adds a profile organization

Usage:


```
geoflow_profile$addLogo(logo)
```

Arguments:

logo logo

Method `setMode()`: Sets profile mode

Usage:

```
geoflow_profile$setMode(mode)
```

Arguments:

mode profile mode

Method `setOption()`: Set global config option

Usage:

```
geoflow_profile$setOption(name, value)
```

Arguments:

name option name

value option value

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_profile$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
profile <- geoflow_profile$new()
profile$setId("workflow1")
profile$setName("Workflow 1")
profile$setProject("My project")
profile$setOrganization("My organization")
profile$addLogo("https://via.placeholder.com/300x150.png/09f/fff?text=geoflow")

## End(Not run)
```

geoflow_provenance	<i>Geoflow provenance class</i>
--------------------	---------------------------------

Description

This class models an provenance

Format

[R6Class](#) object.

Details

geoflow_provenance

Value

Object of [R6Class](#) for modelling an provenance

Public fields

statement provenance statement

processes list of processes, as objects of class [geoflow_process](#)

Methods**Public methods:**

- [geoflow_provenance\\$new\(\)](#)
- [geoflow_provenance\\$setStatement\(\)](#)
- [geoflow_provenance\\$addProcess\(\)](#)
- [geoflow_provenance\\$clone\(\)](#)

Method [new\(\)](#): Initializes a [geoflow_provenance](#)

Usage:

```
geoflow_provenance$new(str = NULL)
```

Arguments:

str character string to initialize a provenance using key-based syntax

Method [setStatement\(\)](#): Set process statement

Usage:

```
geoflow_provenance$setStatement(statement)
```

Arguments:

statement process statement

Method [addProcess\(\)](#): Adds process

Usage:

```
geoflow_provenance$addProcess(process)
```

Arguments:

process, object of class [geoflow_process](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_provenance$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
provenance <- geoflow_provenance$new()
provenance$setStatement("statement")
process1 <- geoflow_process$new()
process1$setRationale("task 1")
process1$setDescription("Performs task 1")
provenance$addProcess(process1)
process2 <- geoflow_process$new()
process2$setRationale("task 2")
process2$setDescription("Performs task 2")
provenance$addProcess(process2)

## End(Not run)
```

geoflow_register

Geoflow register class

Description

This class models a register to be used by geoflow

Format

[R6Class](#) object.

Details

geoflow_register

Value

Object of [R6Class](#) for modelling a register

Public fields

```
id register id
def register def
fun register function
data register data
```

Methods**Public methods:**

- [geoflow_register\\$new\(\)](#)
- [geoflow_register\\$fetch\(\)](#)
- [geoflow_register\\$check\(\)](#)
- [geoflow_register\\$clone\(\)](#)

Method [new\(\)](#): Initializes an object of class [geoflow_register](#)

Usage:

```
geoflow_register$new(id, def, fun)
```

Arguments:

```
id id
def def
fun fun
```

Method [fetch\(\)](#): Fetchs the register data using the register function

Usage:

```
geoflow_register$fetch(config = NULL)
```

Arguments:

```
config a geoflow config object
```

Method [check\(\)](#): Checks the register data structure. The structure of the `data.frame` returned by the register function should be of 4 columns including "code", "uri", "label", "definition". In case the data structure is not valid, the function throws an error.

Usage:

```
geoflow_register$check(data)
```

Arguments:

```
data a register data structure
```

Method [clone\(\)](#): The objects of this class are cloneable with this method.

Usage:

```
geoflow_register$clone(deep = FALSE)
```

Arguments:

```
deep Whether to make a deep clone.
```

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
register <- geoflow_register$new(
  id = "some-id",
  def = "definition",
  fun = function(){}
)

## End(Not run)
```

geoflow_relation	<i>Geoflow relation class</i>
------------------	-------------------------------

Description

This class models an relation

Format

[R6Class](#) object.

Details

geoflow_relation

Value

Object of [R6Class](#) for modelling an relation

Public fields

key relation key
link relation link
mimeType relation mime
name relation name
description relation name
prov provenance,

Methods

Public methods:

- `geoflow_relation$new()`
- `geoflow_relation$setKey()`
- `geoflow_relation$setLink()`
- `geoflow_relation$setMimeType()`
- `geoflow_relation$setName()`
- `geoflow_relation$setDescription()`
- `geoflow_relation$setProv()`
- `geoflow_relation$clone()`

Method `new()`: Initializes an object of class `geoflow_relation`

Usage:

```
geoflow_relation$new(str = NULL)
```

Arguments:

`str` character string to initialize from using key-based syntax

Method `setKey()`: Set key

Usage:

```
geoflow_relation$setKey(key)
```

Arguments:

`key` key

Method `setLink()`: Set link

Usage:

```
geoflow_relation$setLink(link)
```

Arguments:

`link` link

Method `setMimeType()`: Set mime type

Usage:

```
geoflow_relation$setMimeType(mimeType)
```

Arguments:

`mimeType` mime type

Method `setName()`: Set name

Usage:

```
geoflow_relation$setName(name)
```

Arguments:

`name` name

Method `setDescription()`: Set description

Usage:

```
geoflow_relation$setDescription(description)
```

Arguments:

```
description description
```

Method setProv(): Set provenance*Usage:*

```
geoflow_relation$setProv(prov = c("user", "geoflow"))
```

Arguments:

```
prov provenance
```

Method clone(): The objects of this class are cloneable with this method.*Usage:*

```
geoflow_relation$clone(deep = FALSE)
```

Arguments:

```
deep Whether to make a deep clone.
```

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
relation <- geoflow_relation$new()
relation$setProv("user")
relation$setKey("wms")
relation$setLink("http://somelink/wms")
relation$setMimeType("application/xml")
relation$setName("layername")
relation$setDescription("layer description")

## End(Not run)
```

geoflow_right

Geoflow right class

Description

This class models an right

Format

[R6Class](#) object.

Details

geoflow_right

Value

Object of [R6Class](#) for modelling an right

Public fields

key right key

values right values

Methods**Public methods:**

- [geoflow_right\\$new\(\)](#)
- [geoflow_right\\$setKey\(\)](#)
- [geoflow_right\\$setValues\(\)](#)
- [geoflow_right\\$clone\(\)](#)

Method [new\(\)](#): Initializes an object of class [geoflow_right](#)

Usage:

```
geoflow_right$new(str = NULL, kvp = NULL)
```

Arguments:

str character string to initialize from using key-based syntax

kvp an object of class [geoflow_kvp](#)

Method [setKey\(\)](#): Sets key

Usage:

```
geoflow_right$setKey(key)
```

Arguments:

key key

Method [setValues\(\)](#): Sets values

Usage:

```
geoflow_right$setValues(values)
```

Arguments:

values values

Method [clone\(\)](#): The objects of this class are cloneable with this method.

Usage:

```
geoflow_right$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
right <- geoflow_right$new()
right$setKey("use")
right$setValues("No restrictions")

## End(Not run)
```

geoflow_skos_vocabulary

Geoflow SKOS vocabulary class

Description

This class models a SKOS vocabulary

Format

[R6Class](#) object.

Details

geoflow_skos_vocabulary

Value

Object of [R6Class](#) for modelling a SKOS vocabulary

Super class

[geoflow::geoflow_vocabulary](#) -> geoflow_skos_vocabulary

Public fields

rdf rdf
rdf_data rdf_data
endpoint endpoint

Methods

Public methods:

- `geoflow_skos_vocabulary$new()`
- `geoflow_skos_vocabulary$query()`
- `geoflow_skos_vocabulary$query_full_dataset()`
- `geoflow_skos_vocabulary$ping()`
- `geoflow_skos_vocabulary$list_collections()`
- `geoflow_skos_vocabulary$get_concepts_hierarchy()`
- `geoflow_skos_vocabulary$list_concepts()`
- `geoflow_skos_vocabulary$query_from_uri()`
- `geoflow_skos_vocabulary$query_from_term()`
- `geoflow_skos_vocabulary$clone()`

Method `new()`: Initializes a vocabulary

Usage:

```
geoflow_skos_vocabulary$new(id, def, uri, endpoint = NULL, file = NULL)
```

Arguments:

`id` id

`def` def

`uri` uri

`endpoint` A Sparql endpoint

`file` a RDF file

Method `query()`: query

Usage:

```
geoflow_skos_vocabulary$query(str, graphUri = NULL, mimetype = "text/csv")
```

Arguments:

`str` str

`graphUri` graphUri

`mimetype` mimetype

Returns: the response of the SPARQL query

Method `query_full_dataset()`: Queries full dataset

Usage:

```
geoflow_skos_vocabulary$query_full_dataset()
```

Returns: an object of class [tibble](#)

Method `ping()`: Ping query

Usage:

```
geoflow_skos_vocabulary$ping()
```

Method `list_collections()`: list_collections

Usage:

```
geoflow_skos_vocabulary$list_collections(  
  mimetype = "text/csv",  
  count_sub_collections = TRUE,  
  count_concepts = TRUE  
)
```

Arguments:

mimetype mimetype
count_sub_collections count_sub_collections. Default is TRUE
count_concepts count_concepts. Default is TRUE

Returns: the response of the SPARQL query

Method get_concepts_hierarchy(): list_concepts*Usage:*

```
geoflow_skos_vocabulary$get_concepts_hierarchy(  
  lang = "en",  
  method = c("SPARQL", "R"),  
  out_format = c("tibble", "list")  
)
```

Arguments:

lang lang
method method used to build the hierarchy, either "SPARQL" or "R"
out_format output format (tibble or list). Default is "tibble"

Returns: the response of the SPARQL query

Method list_concepts(): list_concepts*Usage:*

```
geoflow_skos_vocabulary$list_concepts(lang = "en", mimetype = "text/csv")
```

Arguments:

lang lang
mimetype mimetype

Returns: the response of the SPARQL query

Method query_from_uri(): query_from_uri*Usage:*

```
geoflow_skos_vocabulary$query_from_uri(  
  uri,  
  graphUri = NULL,  
  mimetype = "text/csv"  
)
```

Arguments:

uri uri
graphUri graphUri

mimetype mimetype

Returns: an object of class [tibble](#)

Method query_from_term(): query_from_term

Usage:

```
geoflow_skos_vocabulary$query_from_term(
  term,
  graphUri = NULL,
  mimetype = "text/csv"
)
```

Arguments:

term term
graphUri graphUri
mimetype mimetype

Returns: an object of class [tibble](#)

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_skos_vocabulary$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

geoflow_software

Geoflow software class

Description

This class models a software to be used by geoflow

Format

[R6Class](#) object.

Details

geoflow_software

Value

Object of [R6Class](#) for modelling a software

Super class

[geoflow::geoflowLogger](#) -> geoflow_software

Public fields

id software id
 type software I/O type ("input" or "output")
 software_type type of software
 definition definition
 packages list of packages required for the software functioning
 handler software handler function
 arguments software arguments
 parameters software parameters
 attributes software attributes
 properties software properties
 actions actions associated with the software

Methods**Public methods:**

- `geoflow_software$new()`
- `geoflow_software$setId()`
- `geoflow_software$setType()`
- `geoflow_software$setSoftwareType()`
- `geoflow_software$setPackages()`
- `geoflow_software$setDefinition()`
- `geoflow_software$setAttributes()`
- `geoflow_software$setProperties()`
- `geoflow_software$setArguments()`
- `geoflow_software$setParameters()`
- `geoflow_software$setActions()`
- `geoflow_software$setHandler()`
- `geoflow_software$checkPackages()`
- `geoflow_software$getHandlerInstance()`
- `geoflow_software$clone()`

Method `new()`: Initializes a software

Usage:

```

geoflow_software$new(
  id = NULL,
  type = NULL,
  software_type,
  packages = list(),
  definition,
  handler,
  arguments,
  attributes = list(),
  actions = list()
)
```

Arguments:

id id
 type type "input" or "output"
 software_type software type
 packages list of packages required for the software functioning
 definition software definition
 handler software handler function
 arguments software handler arguments
 attributes software attributes
 actions software actions

Method setId(): Sets software ID*Usage:*

```
geoflow_software$setId(id)
```

Arguments:

id id

Method setType(): Set type. Either "input" or "output"*Usage:*

```
geoflow_software$setType(type)
```

Arguments:

type software I/O type

Method setSoftwareType(): Set software type*Usage:*

```
geoflow_software$setSoftwareType(software_type)
```

Arguments:

software_type software type

Method setPackages(): Set software required packages*Usage:*

```
geoflow_software$setPackages(packages)
```

Arguments:

packages list of package names

Method setDefinition(): Set software definition*Usage:*

```
geoflow_software$setDefinition(definition)
```

Arguments:

definition software definition

Method setAttributes(): Set attributes. Function to call when creating an instance of geoflow_software

Usage:

```
geoflow_software$setAttributes(attributes)
```

Arguments:

attributes named list of attributes

Method `setProperties()`: Set properties. Function to call to pass argument values for a given `geoflow_software`

Usage:

```
geoflow_software$setProperties(props)
```

Arguments:

props named list of properties

Method `setArguments()`: Set software arguments. Function to call when creating an instance of `geoflow_software`

Usage:

```
geoflow_software$setArguments(arguments)
```

Arguments:

arguments list of software arguments

Method `setParameters()`: Set parameters. Function to call to pass argument values for a given `geoflow_software`

Usage:

```
geoflow_software$setParameters(params)
```

Arguments:

params named list of parameters

Method `setActions()`: Set software actions

Usage:

```
geoflow_software$setActions(actions)
```

Arguments:

actions a list of `geoflow_action`

Method `setHandler()`: Set the software handler function

Usage:

```
geoflow_software$setHandler(handler)
```

Arguments:

handler object of class function

Method `checkPackages()`: Check that all packages required for the software are available, if yes, import them in the R session, and return a `data.frame` giving the packages names and version. If one or more packages are unavailable, an error is thrown and user informed of the missing packages.

Usage:

```
geoflow_software$checkPackages()
```

Method `getHandlerInstance()`: Get the software handler instance

Usage:

```
geoflow_software$getHandlerInstance()
```

Returns: an object instance of the software handler

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_software$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
software<- geoflow_software$new(
  id = "some-id",
  type = "output",
  software_type = "software",
  definition = "definition",
  packages = list(),
  handler = function() {},
  arguments = list(
    url = list(def = "the software url")
  ),
  attributes = list(
    workspace = list(def = "a workspace name in the software")
  )
)

## End(Not run)
```

geoflow_subject

Geoflow subject class

Description

This class models a subject

Format

[R6Class](#) object.

Details

geoflow_subject

Value

Object of [R6Class](#) for modelling a subject

Public fields

key subject key
name subject name
uri subject URI
dates subject date(s)
keywords subject keywords

Methods**Public methods:**

- [geoflow_subject\\$new\(\)](#)
- [geoflow_subject\\$setKey\(\)](#)
- [geoflow_subject\\$setName\(\)](#)
- [geoflow_subject\\$setUri\(\)](#)
- [geoflow_subject\\$setDate\(\)](#)
- [geoflow_subject\\$addKeyword\(\)](#)
- [geoflow_subject\\$getKeywords\(\)](#)
- [geoflow_subject\\$clone\(\)](#)

Method [new\(\)](#): Initializes an object of class [geoflow_subject](#)

Usage:

```
geoflow_subject$new(str = NULL, kvp = NULL)
```

Arguments:

str a character string to initialize from, using key-based syntax

kvp an object of class [geoflow_kvp](#)

Method [setKey\(\)](#): Sets subject key

Usage:

```
geoflow_subject$setKey(key)
```

Arguments:

key key

Method [setName\(\)](#): Sets subject name

Usage:

```
geoflow_subject$setName(name)
```

Arguments:

name name

Method `setUri()`: Sets subject URI

Usage:

```
geoflow_subject$setUri(uri)
```

Arguments:

uri uri

Method `setDate()`: Sets date

Usage:

```
geoflow_subject$setDate(dateType, date)
```

Arguments:

dateType type of date

date date

Method `addKeyword()`: Adds a keyword

Usage:

```
geoflow_subject$addKeyword(keyword, uri = NULL)
```

Arguments:

keyword keyword

uri keyword URI. Default is NULL

Method `getKeywords()`: Get keywords associated with the subject

Usage:

```
geoflow_subject$getKeywords(pretty = FALSE)
```

Arguments:

pretty whether the output has to be prettyfied as data.frame

Returns: the list of keywords as list of geoflow_keyword objects or data.frame

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_subject$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
subject <- geoflow_subject$new()
subject$setKey("theme")
subject$setName("General")
subject$setUri("http://somelink/general")
subject$addKeyword("keyword1", "http://somelink/keyword1")
subject$addKeyword("keyword2", "http://somelink/keyword2")
subject$addKeyword("keyword3", "http://somelink/keyword3")

## End(Not run)
```

geoflow_validator

geoflow_validator

Description

geoflow_validator

geoflow_validator

Public fields

source object of class data.frame handling metadata objects to validate

Methods**Public methods:**

- `geoflow_validator$new()`
- `geoflow_validator$validate_structure()`
- `geoflow_validator$validate_content()`
- `geoflow_validator$clone()`

Method `new()`: Initializes a table validator for a given metadata model

Usage:

```
geoflow_validator$new(model, valid_columns, source)
```

Arguments:

model the data model name, eg. "entity", "contact"

valid_columns a vector of valid columns for the data model

source an object of class data.frame handling the contacts

Method `validate_structure()`: Validates a source table against a data model structure

Usage:

```
geoflow_validator$validate_structure()
```

Returns: TRUE if valid, FALSE otherwise.

Method `validate_content()`: Validates a source table using syntactic and content validation rules

Usage:

```
geoflow_validator$validate_content(raw = FALSE)
```

Arguments:

`raw` indicates whether to return a list of `geoflow_validator_cell` objects or a `data.frame`
`debug` debug validation

Returns: a list of `geoflow_validator_cell` objects, or `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`geoflow_validator_cell`

geoflow_validator_cell

Description

`geoflow_validator_cell`

`geoflow_validator_cell`

Public fields

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

Methods

Public methods:

- `geoflow_validator_cell$new()`
- `geoflow_validator_cell$isNaAuthorized()`
- `geoflow_validator_cell$isKeySynthaxUser()`
- `geoflow_validator_cell$isKeyRequired()`
- `geoflow_validator_cell$getValidKeys()`
- `geoflow_validator_cell$isErrorIfInvalidKey()`
- `geoflow_validator_cell$getDefaultKey()`
- `geoflow_validator_cell$isExcludeHttpKeys()`
- `geoflow_validator_cell$isMultiple()`
- `geoflow_validator_cell$validate()`

- `geoflow_validator_cell$clone()`

Method `new()`: Initializes a `geoflow_validator_cell`

Usage:

```
geoflow_validator_cell$new(
  na_authorized,
  use_key_syntax,
  key_required,
  valid_keys,
  default_key,
  error_if_invalid_key,
  exclude_http_keys,
  multiple,
  i,
  j,
  str
)
```

Arguments:

`na_authorized` if validator cell authorizes NAs or empty strings. Default is FALSE
`use_key_syntax` if validator cell uses key-based syntax. Default is TRUE
`key_required` if validator cell has a key required. Default is TRUE
`valid_keys` valid keys for the validator cell. Default is an empty list
`default_key` default_key to use if key is omitted. Default is NULL
`error_if_invalid_key` raise an error if key is invalid. Default is TRUE
`exclude_http_keys` if 'http' keys have to be excluded from validation checks. Default is TRUE
`multiple` if cell may contain multiple values. Deprecated
`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` character string to validate

Method `isNaAuthorized()`: Indicates if the validator cell authorizes NAs and empty strings

Usage:

```
geoflow_validator_cell$isNaAuthorized()
```

Returns: TRUE if authorizes NAs and empty strings, FALSE otherwise

Method `isKeySynthaxUser()`: Indicates if the validator cell makes use of key-based syntax

Usage:

```
geoflow_validator_cell$isKeySynthaxUser()
```

Returns: TRUE if using key-based syntax, FALSE otherwise

Method `isKeyRequired()`: Indicates if a key is required for the validator cell

Usage:

```
geoflow_validator_cell$isKeyRequired()
```

Returns: TRUE if requires a key, FALSE otherwise

Method getValidKeys(): Gives the list of valid keys for the validator cell

Usage:

```
geoflow_validator_cell$getValidKeys()
```

Returns: the list of valid keys

Method isErrorIfInvalidKey(): Indicates if a report error will be given in case of invalid key

Usage:

```
geoflow_validator_cell$isErrorIfInvalidKey()
```

Returns: TRUE if a report error will be given in case of invalid key, FALSE otherwise

Method getDefaultKey(): Gets the default key

Usage:

```
geoflow_validator_cell$getDefaultKey()
```

Returns: the default key

Method isExcludeHttpKeys(): Indicates if 'http' keys are excluded from the validation

Usage:

```
geoflow_validator_cell$isExcludeHttpKeys()
```

Returns: TRUE if 'http' keys are excluded from the validation, FALSE otherwise

Method isMultiple(): indicates if multiple key-based components can be used within a same cell

Usage:

```
geoflow_validator_cell$isMultiple()
```

Returns: TRUE if supporting multiple key-based components by cell, FALSE otherwise

Method validate(): Proceeds with syntactic validation for the cell considered

Usage:

```
geoflow_validator_cell$validate()
```

Returns: an object of class data.frame including possible errors/warnings

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_cell$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_contacts
      geoflow_validator_contacts
```

Description

```
geoflow_validator_contacts
geoflow_validator_contacts
```

Super class

```
geoflow::geoflow_validator -> geoflow_validator_contacts
```

Methods**Public methods:**

- `geoflow_validator_contacts$new()`
- `geoflow_validator_contacts$clone()`

Method `new()`: Initializes a contacts table validator

Usage:

```
geoflow_validator_contacts$new(source)
```

Arguments:

source an object of class `data.frame` handling the contacts

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_contacts$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_contact_Identifier
      geoflow_validator_contact_Identifier
```

Description

```
geoflow_validator_contact_Identifier
geoflow_validator_contact_Identifier
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_contact_Identifier
```

Methods**Public methods:**

- `geoflow_validator_contact_Identifier$new()`
- `geoflow_validator_contact_Identifier$validate()`
- `geoflow_validator_contact_Identifier$clone()`

Method `new()`: Initializes a contact 'Identifier' cell

Usage:

```
geoflow_validator_contact_Identifier$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `validate()`: Validates a contact identifier. Proceeds with syntactic validation and content (ORCID) validation.

Usage:

```
geoflow_validator_contact_Identifier$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_contact_Identifier$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entities
```

```
geoflow_validator_entities
```

Description

```
geoflow_validator_entities
```

```
geoflow_validator_entities
```

Super class

```
geoflow::geoflow_validator -> geoflow_validator_entities
```


Methods**Public methods:**

- `geoflow_validator_entities$new()`
- `geoflow_validator_entities$clone()`

Method `new()`: Initializes an entities table validator

Usage:

```
geoflow_validator_entities$new(source)
```

Arguments:

source an object of class `data.frame` handling the entities

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entities$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Creator
```

```
  geoflow_validator_entity_Creator
```

Description

```
geoflow_validator_entity_Creator
```

```
geoflow_validator_entity_Creator
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Creator
```

Methods**Public methods:**

- `geoflow_validator_entity_Creator$new()`
- `geoflow_validator_entity_Creator$clone()`

Method `new()`: Initializes an entity 'Creator' cell

Usage:

```
geoflow_validator_entity_Creator$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Creator$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Data
```

```
geoflow_validator_entity_Data
```

Description

```
geoflow_validator_entity_Data
```

```
geoflow_validator_entity_Data
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Data
```

Methods

Public methods:

- `geoflow_validator_entity_Data$new()`
- `geoflow_validator_entity_Data$clone()`

Method new(): Initializes an entity 'Data' cell

Usage:

```
geoflow_validator_entity_Data$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Data$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Date
      geoflow_validator_entity_Date
```

Description

```
geoflow_validator_entity_Date
geoflow_validator_entity_Date
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Date
```

Methods

Public methods:

- `geoflow_validator_entity_Date$new()`
- `geoflow_validator_entity_Date$validate()`
- `geoflow_validator_entity_Date$clone()`

Method `new()`: Initializes an entity 'Date' cell

Usage:

```
geoflow_validator_entity_Date$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)
j col index (internal index to be used for graphical **geoflow** validation handlers)
str string to validate

Method `validate()`: Validates a date. Proceeds with syntactic validation and content validation.

Usage:

```
geoflow_validator_entity_Date$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Date$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

geoflow_validator_entity_Description
geoflow_validator_entity_Description

Description

geoflow_validator_entity_Description

geoflow_validator_entity_Description

Super class

geoflow::geoflow_validator_cell -> geoflow_validator_entity_Description

Methods

Public methods:

- `geoflow_validator_entity_Description$new()`
- `geoflow_validator_entity_Description$clone()`

Method `new()`: Initializes an entity 'Description' cell

Usage:

`geoflow_validator_entity_Description$new(i, j, str)`

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`geoflow_validator_entity_Description$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Format
    geoflow_validator_entity_Format
```

Description

geoflow_validator_entity_Format

geoflow_validator_entity_Format

Super class

`geoflow::geoflow_validator_cell` -> geoflow_validator_entity_Format

Methods

Public methods:

- `geoflow_validator_entity_Format$new()`
- `geoflow_validator_entity_Format$clone()`

Method `new()`: Initializes an entity 'Format' cell

Usage:

```
geoflow_validator_entity_Format$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Format$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Identifier
    geoflow_validator_entity_Identifier
```

Description

```
geoflow_validator_entity_Identifier
geoflow_validator_entity_Identifier
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Identifier
```

Methods

Public methods:

- `geoflow_validator_entity_Identifier$new()`
- `geoflow_validator_entity_Identifier$validate()`
- `geoflow_validator_entity_Identifier$clone()`

Method `new()`: Initializes an entity 'Identifier' cell

Usage:

```
geoflow_validator_entity_Identifier$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates an entity identifier. Proceeds with syntactic validation and content validation for DOIs and UUIDs.

Usage:

```
geoflow_validator_entity_Identifier$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Identifier$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_Language
    geoflow_validator_entity_Language
```

Description

```
geoflow_validator_entity_Language
geoflow_validator_entity_Language
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Language
```

Methods**Public methods:**

- `geoflow_validator_entity_Language$new()`
- `geoflow_validator_entity_Language$validate()`
- `geoflow_validator_entity_Language$clone()`

Method `new()`: Initializes an entity 'Language' cell

Usage:

```
geoflow_validator_entity_Language$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates a language. Proceeds with syntactic validation and language validation.

Usage:

```
geoflow_validator_entity_Language$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Language$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_Provenance
    geoflow_validator_entity_Provenance
```

Description

```
geoflow_validator_entity_Provenance
geoflow_validator_entity_Provenance
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Provenance
```

Methods

Public methods:

- `geoflow_validator_entity_Provenance$new()`
- `geoflow_validator_entity_Provenance$validate()`
- `geoflow_validator_entity_Provenance$clone()`

Method `new()`: Initializes an entity 'Provenance' cell

Usage:

```
geoflow_validator_entity_Provenance$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates a Provenance. Proceeds with syntactic validation and content validation.

Usage:

```
geoflow_validator_entity_Provenance$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Provenance$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_Relation
    geoflow_validator_entity_Relation
```

Description

geoflow_validator_entity_Relation

geoflow_validator_entity_Relation

Super class

`geoflow::geoflow_validator_cell` -> geoflow_validator_entity_Relation

Methods

Public methods:

- `geoflow_validator_entity_Relation$new()`
- `geoflow_validator_entity_Relation$clone()`

Method `new()`: Initializes an entity 'Relation' cell

Usage:

```
geoflow_validator_entity_Relation$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Relation$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_Rights
      geoflow_validator_entity_Rights
```

Description

```
geoflow_validator_entity_Rights
```

```
geoflow_validator_entity_Rights
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Rights
```

Methods

Public methods:

- `geoflow_validator_entity_Rights$new()`
- `geoflow_validator_entity_Rights$clone()`

Method `new()`: Initializes an entity 'Rights' cell

Usage:

```
geoflow_validator_entity_Rights$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Rights$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

```
geoflow_validator_entity_SpatialCoverage
    geoflow_validator_entity_SpatialCoverage
```

Description

```
geoflow_validator_entity_SpatialCoverage
geoflow_validator_entity_SpatialCoverage
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_SpatialCoverage
```

Methods

Public methods:

- `geoflow_validator_entity_SpatialCoverage$new()`
- `geoflow_validator_entity_SpatialCoverage$validate()`
- `geoflow_validator_entity_SpatialCoverage$clone()`

Method `new()`: Initializes an entity 'SpatialCoverage' cell

Usage:

```
geoflow_validator_entity_SpatialCoverage$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates a spatial coverage. Proceeds with syntactic validation and spatial coverage validation (including EWKT, WKT and SRID).

Usage:

```
geoflow_validator_entity_SpatialCoverage$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_SpatialCoverage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_Subject
    geoflow_validator_entity_Subject
```

Description

```
geoflow_validator_entity_Subject
geoflow_validator_entity_Subject
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Subject
```

Methods

Public methods:

- `geoflow_validator_entity_Subject$new()`
- `geoflow_validator_entity_Subject$validate()`
- `geoflow_validator_entity_Subject$clone()`

Method `new()`: Initializes an entity 'Subject' cell

Usage:

```
geoflow_validator_entity_Subject$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates a Subject. Proceeds with syntactic validation and content validation.

Usage:

```
geoflow_validator_entity_Subject$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Subject$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_TemporalCoverage
    geoflow_validator_entity_TemporalCoverage
```

Description

```
geoflow_validator_entity_TemporalCoverage
geoflow_validator_entity_TemporalCoverage
```

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_TemporalCoverage
```

Methods**Public methods:**

- `geoflow_validator_entity_TemporalCoverage$new()`
- `geoflow_validator_entity_TemporalCoverage$validate()`
- `geoflow_validator_entity_TemporalCoverage$clone()`

Method `new()`: Initializes an entity 'TemporalCoverage' cell

Usage:

```
geoflow_validator_entity_TemporalCoverage$new(i, j, str)
```

Arguments:

`i` row index (internal index to be used for graphical **geoflow** validation handlers)
`j` col index (internal index to be used for graphical **geoflow** validation handlers)
`str` string to validate

Method `validate()`: Validates temporal coverage. Proceeds with syntactic validation and temporal coverage validation.

Usage:

```
geoflow_validator_entity_TemporalCoverage$validate()
```

Returns: an validation report, as object of class `data.frame`

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_TemporalCoverage$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

```
geoflow_validator_entity_Title
    geoflow_validator_entity_Title
```

Description

geoflow_validator_entity_Title

geoflow_validator_entity_Title

Super class

```
geoflow::geoflow_validator_cell -> geoflow_validator_entity_Title
```

Methods

Public methods:

- `geoflow_validator_entity_Title$new()`
- `geoflow_validator_entity_Title$clone()`

Method `new()`: Initializes an entity 'Title' cell

Usage:

```
geoflow_validator_entity_Title$new(i, j, str)
```

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
geoflow_validator_entity_Title$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

geoflow_validator_entity_Type
geoflow_validator_entity_Type

Description

geoflow_validator_entity_Type

geoflow_validator_entity_Type

Super class

`geoflow::geoflow_validator_cell` -> geoflow_validator_entity_Type

Methods

Public methods:

- `geoflow_validator_entity_Type$new()`
- `geoflow_validator_entity_Type$clone()`

Method `new()`: Initializes an entity 'Type' cell

Usage:

`geoflow_validator_entity_Type$new(i, j, str)`

Arguments:

i row index (internal index to be used for graphical **geoflow** validation handlers)

j col index (internal index to be used for graphical **geoflow** validation handlers)

str string to validate

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

`geoflow_validator_entity_Type$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

geoflow_vocabulary	<i>Geoflow vocabulary class</i>
--------------------	---------------------------------

Description

This class models a vocabulary

Format

[R6Class](#) object.

Details

geoflow_vocabulary

Value

Object of [R6Class](#) for modelling a vocabulary

Public fields

id id
def def
uri uri
software_type software_type
software software

Methods**Public methods:**

- [geoflow_vocabulary\\$new\(\)](#)
- [geoflow_vocabulary\\$setSoftware\(\)](#)
- [geoflow_vocabulary\\$clone\(\)](#)

Method [new\(\)](#): Initializes a vocabulary

Usage:

[geoflow_vocabulary\\$new](#)(id, def, uri, software_type)

Arguments:

id id
def def
uri uri
software_type software type

Method [setSoftware\(\)](#): Set software

Usage:

```
geoflow_vocabulary$setSoftware(software)
```

Arguments:

```
software software
```

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
geoflow_vocabulary$clone(deep = FALSE)
```

Arguments:

```
deep Whether to make a deep clone.
```

```
getDBTableColumnComment
```

```
getDBTableColumnComment
```

Description

```
getDBTableColumnComment
```

Usage

```
getDBTableColumnComment(dbi, schema, table, column_index)
```

Arguments

dbi	a dbi connection
schema	schema
table	table
column_index	table column index

Value

```
the table comment
```

Author(s)

```
Emmanuel Blondel, <emmanuel.blondel1@gmail.com>
```

getDBTableComment	<i>getDBTableComment</i>
-------------------	--------------------------

Description

getDBTableComment

Usage

```
getDBTableComment(dbi, schema, table)
```

Arguments

dbi	a dbi connection
schema	schema
table	table

Value

the table comment

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

get_absolute_path	<i>get_absolute_path</i>
-------------------	--------------------------

Description

get_absolute_path allows to get the absolute path of a resource given a base directory

Usage

```
get_absolute_path(path, base, mustWork, expand_tilde)
```

Arguments

path	a path in character string
base	a base direcotry
mustWork	must work?
expand_tilde	expand tilde?

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

get_config_resource_path
get_config_resource_path

Description

get_config_resource_path

Usage

get_config_resource_path(config, path)

Arguments

config	a geoflow config
path	a resource path to resolve vs. the config root dir

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

get_epsg_code *get_epsg_code*

Description

get_epsg_code is a consolidated method to get EPSG code (srid) from a CRS

Usage

get_epsg_code(x)

Arguments

x	an object of class 'sf'
---	-------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

get_line_separator	<i>get_line_separator</i>
--------------------	---------------------------

Description

get_line_separator get the line separator used by geoflow when extracting cell components for tabular data content handling. Default is set to an underscore followed by a line break.

Usage

```
get_line_separator()
```

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

get_locales_from	<i>get_locales_from</i>
------------------	-------------------------

Description

Get locales from a property values set

Usage

```
get_locales_from(values)
```

Arguments

values	values
--------	--------

get_union_bbox	<i>get_union_bbox</i>
----------------	-----------------------

Description

get_union_bbox will build a unified bounding box from a list of geoflow_data objects

Usage

```
get_union_bbox(data_objects)
```

Arguments

data_objects	list of geoflow_data objects
--------------	------------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

initWorkflow	<i>initWorkflow</i>
--------------	---------------------

Description

initWorkflow allows to init a workflow

Usage

```
initWorkflow(file, dir, jobDirPath, handleMetadata, session)
```

Arguments

file	a JSON or YAML configuration file
dir	a directory where to execute the workflow.
jobDirPath	a directory set-up for the job. Default is NULL means it will be created during initialization of the workflow, otherwise the path provided will be used.
handleMetadata	Default is TRUE. Metadata contacts/entities/dictionary will be handled. If set to FALSE, they will not be handled. This is used for example in geoflow Shiny app where we want to initialize config without handling metadata to inherit software connections and test dynamically the metadata validity.
session	a shiny session object (optional) to run geoflow in a shiny context

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

initWorkflowJob	<i>initWorkflowJob</i>
-----------------	------------------------

Description

initWorkflowJob allows to init a workflow job

Usage

```
initWorkflowJob(dir)
```

Arguments

dir	a directory where to initialize/execute the workflow
-----	--

Value

the job directory path

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

<code>is_absolute_path</code>	<i>is_absolute_path</i>
-------------------------------	-------------------------

Description

`is_absolute_path` evaluate if a `{{path}}` expression is an absolute path, the function will return a boolean argument.

Usage

`is_absolute_path(path)`

Arguments

`path` a path in character string

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

<code>list_actions</code>	<i>list_actions</i>
---------------------------	---------------------

Description

`list_actions` lists the actions supported by geoflow.

Usage

`list_actions(raw)`

Arguments

`raw` Default value is FALSE, meaning the actions will be listed as `data.frame`. The output If TRUE the raw list of [geoflow_action](#) is returned.

Value

an object of class `data.frame` (or list of [geoflow_action](#) if `raw = FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_action_options *list_action_options*

Description

list_action_options lists the options available for a given action supported by geoflow.

Usage

```
list_action_options(id, raw)
```

Arguments

id	An action identifier
raw	if raw list should be returned

Value

an object of class data.frame (or list if raw is TRUE) listing the available action options.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_contact_handlers *list_contact_handlers*

Description

list_contact_handlers lists the contact handlers supported by geoflow.

Usage

```
list_contact_handlers(raw)
```

Arguments

raw	Default value is FALSE), meaning the handlers will be listed as data.frame. The output If TRUE the raw list of geoflow_handler is returned.
-----	---

Value

an object of class data.frame (or list of [geoflow_handler](#) if raw = FALSE)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_contact_handler_options
list_contact_handler_options

Description

list_contact_handler_options lists the options available for a given contact handler supported by geoflow.

Usage

```
list_contact_handler_options(id, raw)
```

Arguments

id	An contact handler identifier
raw	if raw list should be returned

Value

an object of class data.frame (or list if raw is TRUE) listing the available handler options.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_data_accessors *list_data_accessors*

Description

list_data_accessors lists the data accessors supported by geoflow.

Usage

```
list_data_accessors(raw)
```

Arguments

raw	Default value is FALSE, meaning the data accessors will be listed as data.frame. The output If TRUE the raw list of geoflow_data_accessor is returned.
-----	--

Value

an object of class `data.frame` (or list of [geoflow_data_accessor](#) if `raw = FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
list_dictionary_handlers
      list_dictionary_handlers
```

Description

`list_dictionary_handlers` lists the dictionary handlers supported by geoflow.

Usage

```
list_dictionary_handlers(raw)
```

Arguments

<code>raw</code>	Default value is <code>FALSE</code> , meaning the handlers will be listed as <code>data.frame</code> . The output If <code>TRUE</code> the raw list of geoflow_handler is returned.
------------------	---

Value

an object of class `data.frame` (or list of [geoflow_handler](#) if `raw = FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
list_dictionary_handler_options
      list_dictionary_handler_options
```

Description

`list_dictionary_handler_options` lists the options available for a given dictionary handler supported by geoflow.

Usage

```
list_dictionary_handler_options(id, raw)
```

Arguments

id	An dictionary handler identifier
raw	if raw list should be returned

Value

an object of class `data.frame` (or `list` if `raw` is `TRUE`) listing the available handler options.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_entity_handlers *list_entity_handlers*

Description

`list_entity_handlers` lists the entity handlers supported by geoflow.

Usage

```
list_entity_handlers(raw)
```

Arguments

raw	Default value is <code>FALSE</code> , meaning the handlers will be listed as <code>data.frame</code> . The output If <code>TRUE</code> the raw list of geoflow_handler is returned.
-----	---

Value

an object of class `data.frame` (or `list` of [geoflow_handler](#) if `raw` = `FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_entity_handler_options	
	<i>list_entity_handler_options</i>

Description

list_entity_handler_options lists the options available for a given entity handler supported by geoflow.

Usage

```
list_entity_handler_options(id, raw)
```

Arguments

id	An entity handler identifier
raw	if raw list should be returned

Value

an object of class `data.frame` (or `list` if `raw` is `TRUE`) listing the available handler options.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

list_registers	<i>list_registers</i>
----------------	-----------------------

Description

list_registers lists the registers supported by geoflow.

Usage

```
list_registers(raw)
```

Arguments

raw	Default value is <code>FALSE</code> , meaning the registers will be listed as <code>data.frame</code> . The output If <code>TRUE</code> the raw list of geoflow_register is returned.
-----	---

Value

an object of class `data.frame` (or `list` of [geoflow_register](#) if `raw = FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

<code>list_software</code>	<i>list_software</i>
----------------------------	----------------------

Description

`list_software` lists the software supported by geoflow.

Usage

`list_software(raw)`

Arguments

`raw` Default value is FALSE, meaning the software will be listed as `data.frame`. The output If TRUE the raw list of [geoflow_software](#) is returned.

Value

an object of class `data.frame` (or list of [geoflow_software](#) if `raw = FALSE`)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

<code>list_software_parameters</code>
<i>list_software_parameters</i>

Description

`list_software_parameters` lists the parameters of a given software supported by geoflow.

Usage

`list_software_parameters(software_type, raw)`

Arguments

`software_type` A software type
`raw` if raw list should be returned

Value

an object of class `data.frame` (or `list` if `raw` is `TRUE`) listing the software parameters

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
list_software_properties  
  list_software_properties
```

Description

`list_software_properties` lists the properties of a given software supported by geoflow.

Usage

```
list_software_properties(software_type, raw)
```

Arguments

<code>software_type</code>	A software type
<code>raw</code>	if raw list should be returned

Value

an object of class `data.frame` listing the software properties

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
list_vocabularies    list_vocabularies
```

Description

`list_vocabularies` lists the vocabularies supported by geoflow.

Usage

```
list_vocabularies(raw)
```

Arguments

raw Default value is FALSE, meaning the vocabularies will be listed as `data.frame`.
The output If TRUE the raw list of [geoflow_vocabulary](#) is returned.

Value

an object of class `data.frame` (or list of [geoflow_vocabulary](#) if `raw = FALSE`)

loadMetadataHandler	<i>loadMetadataHandler</i>
---------------------	----------------------------

Description

loadMetadataHandler allows to load a metadata handler

Usage

```
loadMetadataHandler(config, element, type)
```

Arguments

config a geoflow configuration (as list). Only used to write logs, can be NULL.
element a geoflow configuration metadata list element
type either 'contacts', 'entities' or 'dictionary'

Value

an object of class [geoflow_handler](#)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

load_workflow_environment	<i>load_workflow_environment</i>
---------------------------	----------------------------------

Description

load_workflow_environment loads a workflow environment by evaluating variable expressions in the form `${variable}`. If no variable expression pattern is identified in the string, the function will return the original string.

Usage

```
load_workflow_environment(config, session)
```

Arguments

config	object of class list
session	a shiny session object (optional) to run geoflow in a shiny context.

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

posix_to_str	<i>posix_to_str</i>
--------------	---------------------

Description

posix_to_str converts a POSIX object to ISO string

Usage

```
posix_to_str(posix)
```

Arguments

posix	a POSIX object
-------	----------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

precompute_relationships	<i>precompute_relationships</i>
--------------------------	---------------------------------

Description

precompute_relationships

Usage

```
precompute_relationships(data, parent_key, child_key, child_label)
```

Arguments

data	data
parent_key	parent_key
child_key	child_key
child_label	child_label

Value

a list of relationships

<code>register_actions</code>	<i>register_actions</i>
-------------------------------	-------------------------

Description

`register_actions` registers default geoflow actions

Usage

`register_actions()`

Note

Function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

<code>register_contact_handlers</code>	<i>register_contact_handlers</i>
--	----------------------------------

Description

`register_contact_handlers` registers the default contact handlers for geoflow

Usage

`register_contact_handlers()`

Note

Internal function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
register_data_accessors
    register_data_accessors
```

Description

register_data_accessors registers default geoflow data accessors

Usage

```
register_data_accessors()
```

Note

Function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
register_dictionary_handlers
    register_dictionary_handlers
```

Description

register_dictionary_handlers registers the default dictionary handlers for geoflow

Usage

```
register_dictionary_handlers()
```

Note

Internal function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

register_entity_handlers
register_entity_handlers

Description

register_entity_handlers registers the default entity handlers for geoflow

Usage

register_entity_handlers()

Note

Internal function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

register_registers *register_registers*

Description

register_registers registers default geoflow registers

Usage

register_registers()

Note

Function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

register_software	<i>register_software</i>
-------------------	--------------------------

Description

register_software registers default geoflow software

Usage

```
register_software()
```

Note

Function called on load by geoflow

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

register_vocabularies	<i>register_vocabularies</i>
-----------------------	------------------------------

Description

register_vocabularies registers default geoflow vocabularies

Usage

```
register_vocabularies()
```

Note

Function called on load by geoflow

sanitize_date	<i>sanitize_date</i>
---------------	----------------------

Description

sanitize_date sanitizes a date in geoflow

Usage

```
sanitize_date(date)
```

Arguments

date	an object o class character
------	-----------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

sanitize_str	<i>sanitize_str</i>
--------------	---------------------

Description

sanitize_str sanitizes a string definition in geoflow

Usage

```
sanitize_str(str)
```

Arguments

str	a string as object of class character
-----	---------------------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

`set_i18n`*set_i18n*

Description

Set default locales to a property values set

Usage

```
set_i18n(term_key, default, expr, ...)
```

Arguments

term_key	term key
default	default
expr	expr
...	named values to be passed to expr

`set_line_separator`*set_line_separator*

Description

`set_line_separator` set the line separator to be used by geoflow when extracting cell components for tabular data content handling.

Usage

```
set_line_separator(x)
```

Arguments

x	a string as object of class character representing the line separator.
---	--

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

set_locales_to	<i>set_locales_to</i>
----------------	-----------------------

Description

Set locales to a property values set

Usage

set_locales_to(values, locales)

Arguments

values	values
locales	locales

str_to_posix	<i>str_to_posix</i>
--------------	---------------------

Description

str_to_posix parses a string into a POSIX object

Usage

str_to_posix(str)

Arguments

str	a string as object of class character
-----	---------------------------------------

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
unload_workflow_environment
    unload_workflow_environment
```

Description

unload_workflow_environment unloads a workflow environment, in the case environment was provided by means of a dotenv file, and loaded using **dotenv** by **geoflow**. The function will recover the session environment variables values (useful in case an environment variable was overwritten for the workflow execution).

Usage

```
unload_workflow_environment(config)
```

Arguments

config object of class [list](#)

Author(s)

Emmanuel Blondel, <emmanuel.blondel1@gmail.com>

```
writeWorkflowJobDataResource
    writeWorkflowJobDataResource
```

Description

writeWorkflowJobDataResource allows to transform datasource into different formats

Usage

```
writeWorkflowJobDataResource(entity, config, obj,
                             useFeatures, resourcename, useUploadSource,
                             createIndexes, overwrite, append, chunk.size,
                             type)
```

Arguments

entity a entity object as read by geoflow_entity
 config a configuration object as read by initWorkflow
 obj a sf file
 useFeatures a boolean condition to define if features must be attach to obj file

resourcename	name of data input
useUploadSource	a boolean condition to define if resourcename is same as uploadSource information
createIndexes	a boolean condition for possibility to create indexes for each column
overwrite	a boolean condition for writing to DB. Default is TRUE
append	a boolean condition for appending to DB existing table. Default is FALSE
chunk.size	an object of class integer giving the size of chunks to apply for DB upload. Default is equal to 0L, meaning DB upload will be done without chunking.
type	format to convert into. Formats supported: shp, csv, gpkg, parquet, dbtable

Author(s)

Alexandre Bennici, <bennicialexandre@gmail.com>

Index

- * **accessor**
 - geoflow_data_accessor, 37
- * **access**
 - geoflow_data_accessor, 37
- * **action**
 - geoflow_action, 16
- * **contact**
 - geoflow_contact, 20
 - geoflow_dictionary, 42
 - geoflow_featuremember, 58
 - geoflow_featuretype, 60
- * **data**
 - geoflow_data, 26
 - geoflow_data_accessor, 37
- * **date**
 - geoflow_date, 40
- * **dimension**
 - geoflow_dimension, 44
- * **entity**
 - geoflow_entity, 46
- * **format**
 - geoflow_format, 61
- * **handler**
 - geoflow_handler, 63
- * **keyword**
 - geoflow_keyword, 66
- * **kvp**
 - geoflow_kvp, 68
- * **logger**
 - geoflowLogger, 14
- * **mimetype**
 - geoflow_format, 61
- * **mime**
 - geoflow_format, 61
- * **process**
 - geoflow_process, 69
- * **profile**
 - geoflow_profile, 71
- * **provenance**
 - geoflow_provenance, 74
- * **registers**
 - geoflow_register, 75
- * **relation**
 - geoflow_relation, 77
- * **right**
 - geoflow_right, 79
- * **skos**
 - geoflow_skos_vocabulary, 81
- * **software**
 - geoflow_software, 84
- * **subject**
 - geoflow_subject, 88
- * **vocabulary**
 - geoflow_skos_vocabulary, 81
 - geoflow_vocabulary, 112
- add_config_logger, 5
- build_hierarchical_list, 5
- check_packages, 6
- closeWorkflow, 6
- create_geoflow_data_from_dbi, 7
- create_object_identification_id, 7
- data.frame, 59, 61
- debugWorkflow, 8
- describeOGCRelation, 8
- enrich_text_from_entity, 9
- executeWorkflow, 10
- executeWorkflowJob, 11
- extract_cell_components, 11
- extract_kvp, 12
- extract_kvps, 12
- fetch_layer_styles_from_dbi, 13
- filter_sf_by_cqlfilter, 13
- geoflow, 14

- geoflow-package (geoflow), 14
- geoflow::geoflow_validator, 95, 96
- geoflow::geoflow_validator_cell, 95, 97–111
- geoflow::geoflow_vocabulary, 81
- geoflow::geoflowLogger, 17, 37, 84
- geoflow_action, 16, 17, 36, 118
- geoflow_contact, 20, 22, 25, 50, 70
- geoflow_data, 26, 26, 28, 52
- geoflow_data_accessor, 37, 120, 121
- geoflow_date, 40, 41
- geoflow_dictionary, 35, 42, 42
- geoflow_dimension, 34, 44, 44
- geoflow_entity, 13, 20, 46, 48, 57
- geoflow_featuremember, 58, 58, 61
- geoflow_featuretype, 43, 60, 60
- geoflow_format, 50, 61, 62
- geoflow_handler, 63, 64, 119, 121, 122, 126
- geoflow_keyword, 66, 67
- geoflow_kvp, 68, 80, 89
- geoflow_process, 69, 70, 74, 75
- geoflow_profile, 71, 72
- geoflow_provenance, 52, 74, 74
- geoflow_register, 43, 75, 76, 123
- geoflow_relation, 51, 77, 78
- geoflow_right, 51, 79, 80
- geoflow_skos_vocabulary, 81
- geoflow_software, 84, 124
- geoflow_subject, 50, 88, 89
- geoflow_validator, 91
- geoflow_validator_cell, 92, 93
- geoflow_validator_contact_Identifier, 95
- geoflow_validator_contacts, 95
- geoflow_validator_entities, 96
- geoflow_validator_entity_Creator, 97
- geoflow_validator_entity_Data, 98
- geoflow_validator_entity_Date, 99
- geoflow_validator_entity_Description, 100
- geoflow_validator_entity_Format, 101
- geoflow_validator_entity_Identifier, 102
- geoflow_validator_entity_Language, 103
- geoflow_validator_entity_Provenance, 104
- geoflow_validator_entity_Relation, 105
- geoflow_validator_entity_Rights, 106
- geoflow_validator_entity_SpatialCoverage, 107
- geoflow_validator_entity_Subject, 108
- geoflow_validator_entity_TemporalCoverage, 109
- geoflow_validator_entity_Title, 110
- geoflow_validator_entity_Type, 111
- geoflow_vocabulary, 112, 126
- geoflowLogger, 5, 14, 16
- get_absolute_path, 114
- get_config_resource_path, 115
- get_epsg_code, 115
- get_line_separator, 116
- get_locales_from, 116
- get_union_bbox, 116
- getDBTableColumnComment, 113
- getDBTableComment, 114
- initWorkflow, 117
- initWorkflowJob, 117
- is_absolute_path, 118
- list, 5, 127, 135
- list_action_options, 119
- list_actions, 118
- list_contact_handler_options, 120
- list_contact_handlers, 119
- list_data_accessors, 120
- list_dictionary_handler_options, 121
- list_dictionary_handlers, 121
- list_entity_handler_options, 123
- list_entity_handlers, 122
- list_registers, 123
- list_software, 124
- list_software_parameters, 124
- list_software_properties, 125
- list_vocabularies, 125
- load_workflow_environment, 126
- loadMetadataHandler, 126
- posix_to_str, 127
- precompute_relationships, 127
- R6Class, 14–16, 21, 26, 37, 40, 42, 44, 46, 58, 60–64, 66, 68, 69, 71, 74–77, 79–81, 84, 88, 89, 112
- register_actions, 128
- register_contact_handlers, 128
- register_data_accessors, 129

- register_dictionary_handlers, [129](#)
- register_entity_handlers, [130](#)
- register_registers, [130](#)
- register_software, [131](#)
- register_vocabularies, [131](#)

- sanitize_date, [132](#)
- sanitize_str, [132](#)
- set_i18n, [133](#)
- set_line_separator, [133](#)
- set_locales_to, [134](#)
- sprintf, [15](#), [16](#)
- str_to_posix, [134](#)

- tibble, [82](#), [84](#)

- unload_workflow_environment, [135](#)

- writeWorkflowJobDataResource, [135](#)