

Package ‘atom4R’

August 19, 2022

Version 0.3-1

Date 2022-08-18

Title Tools to Handle and Publish Metadata as 'Atom' XML Format

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

Depends R (>= 3.3), methods

Imports R6, jsonlite, readr, XML, httr, zip, rdfliib, keyring

Suggests testthat

Description Provides tools to read/write/publish metadata based on the 'Atom' XML syndication format. This includes support of 'Dublin Core' XML implementation, and a client to API(s) implementing the 'Atom-Pub' 'SWORD' API specification.

License MIT + file LICENSE

URL <https://github.com/eblondel/atom4R>

BugReports <https://github.com/eblondel/atom4R>

RoxygenNote 7.2.1

NeedsCompilation no

Author Emmanuel Blondel [aut, cre] (<<https://orcid.org/0000-0002-5870-5762>>)

Repository CRAN

Date/Publication 2022-08-19 09:20:34 UTC

R topics documented:

atom4R	3
atom4RLogger	4
AtomAbstractObject	6
AtomAuthor	11
AtomCategory	12
AtomContributor	14
AtomEntry	15
AtomFeed	20

AtomLink	27
AtomNamespace	30
AtomPerson	31
AtomPubClient	33
DCAbstract	36
DCAccessRights	37
DCAccrualMethod	38
DCAccrualPeriodicity	39
DCAccrualPolicy	40
DCAlternative	41
DCAudience	42
DCAvailable	43
DCBibliographicCitation	44
DCConformsTo	45
DCCContributor	46
DCCoverage	47
DCCreated	48
DCCreator	49
DCDate	50
DCDateAccepted	51
DCDateCopyrighted	52
DCDateSubmitted	53
DCDescription	54
DCEducationalLevel	55
DCElement	56
DCEntry	57
DCExtent	94
DCFormat	95
DCHasPart	96
DCHasVersion	97
DCIdentifier	98
DCInstructionalMethod	99
DCIsPartOf	100
DCIsReferencedBy	101
DCIsReplacedBy	102
DCIsRequiredBy	103
DCIssued	104
DCIsVersionOf	105
DCLanguage	106
DCLicense	107
DCMediator	108
DCMedium	109
DCMIVocabulary	110
DCModified	111
DCProvenance	112
DCPublisher	113
DCReferences	114
DCRelation	115

DCReplaces	116
DCRequires	117
DCRights	118
DCRightsHolder	119
DCSource	120
DCSpatial	121
DCSubject	122
DCTableOfContents	123
DCTemporal	124
DCTitle	125
DCType	126
DCValid	127
getAtomClasses	128
getAtomNamespace	128
getAtomNamespaces	129
getAtomSchemas	129
getClassesInheriting	130
getDCMIVocabularies	130
getDCMIVocabulary	131
readDCEntry	131
registerAtomNamespace	132
registerAtomSchema	133
setAtomNamespaces	133
setAtomSchemas	133
setDCMIVocabularies	134
SwordClient	134
SwordDataverseClient	136
SwordHalClient	140
SwordServiceDocument	142

Index	144
--------------	------------

atom4R

Tools to Handle and Publish Metadata as Atom XML Format

Description

Provides tools to read/write/publish metadata based on the Atom XML syndication format. This includes support of Dublin Core XML implementation, and a client to APIs implementing the AtomPub SWORD API specification.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

atom4RLogger

atom4RLogger

Description

atom4RLogger

atom4RLogger

Format

[R6Class](#) object.

Value

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

Public fields

`verbose.info` If package info log messages have to be printed out

`verbose.debug` If curl debug log messages have to be printed out

`loggerType` the type of logger

Methods

Public methods:

- [atom4RLogger\\$logger\(\)](#)
- [atom4RLogger\\$INFO\(\)](#)
- [atom4RLogger\\$WARN\(\)](#)
- [atom4RLogger\\$ERROR\(\)](#)
- [atom4RLogger\\$new\(\)](#)
- [atom4RLogger\\$getClassName\(\)](#)
- [atom4RLogger\\$getClass\(\)](#)
- [atom4RLogger\\$clone\(\)](#)

Method `logger()`: Provides log messages

Usage:

```
atom4RLogger$logger(type, text)
```

Arguments:

`type` type of log ("INFO", "WARN", "ERROR")

`text` the log message text

Method `INFO()`: Provides INFO log messages

Usage:

```
atom4RLogger$INFO(text)
```

Arguments:

text the log message text

Method WARN(): Provides WARN log messages

Usage:

```
atom4RLogger$WARN(text)
```

Arguments:

text the log message text

Method ERROR(): Provides ERROR log messages

Usage:

```
atom4RLogger$ERROR(text)
```

Arguments:

text the log message text

Method new(): Initializes the logger

Usage:

```
atom4RLogger$new(logger = NULL)
```

Arguments:

logger logger type "INFO", "DEBUG" or NULL

Method getClassName(): Get class name

Usage:

```
atom4RLogger$getClassName()
```

Returns: object of class data.frame

Method getClass(): Get class

Usage:

```
atom4RLogger$getClass()
```

Returns: object of class [R6Class](#)

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

Usage:

```
atom4RLogger$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Note

Logger class used internally by atom4R

AtomAbstractObject *Atom feed class*

Description

This class models an atom abstract object

Format

[R6Class](#) object.

Details

AtomAbstractObject

Value

Object of [R6Class](#) for modelling an Atom abstract Object

Super class

[atom4R::atom4RLogger](#) -> AtomAbstractObject

Public fields

wrap wrapping XML element

element element

namespace namespace

defaults defaults

attrs attrs

printAttrs attrs to print

parentAttrs parent attrs

Methods

Public methods:

- [AtomAbstractObject\\$new\(\)](#)
- [AtomAbstractObject\\$setIsDocument\(\)](#)
- [AtomAbstractObject\\$isDocument\(\)](#)
- [AtomAbstractObject\\$getRootElement\(\)](#)
- [AtomAbstractObject\\$getNamespace\(\)](#)
- [AtomAbstractObject\\$createElement\(\)](#)
- [AtomAbstractObject\\$addListElement\(\)](#)
- [AtomAbstractObject\\$delListElement\(\)](#)

- [AtomAbstractObject\\$contains\(\)](#)
- [AtomAbstractObject\\$print\(\)](#)
- [AtomAbstractObject\\$decode\(\)](#)
- [AtomAbstractObject\\$encode\(\)](#)
- [AtomAbstractObject\\$validate\(\)](#)
- [AtomAbstractObject\\$save\(\)](#)
- [AtomAbstractObject\\$isFieldInheritedFrom\(\)](#)
- [AtomAbstractObject\\$getClassName\(\)](#)
- [AtomAbstractObject\\$getClass\(\)](#)
- [AtomAbstractObject\\$getNamespaceDefinition\(\)](#)
- [AtomAbstractObject\\$getXmlElement\(\)](#)
- [AtomAbstractObject\\$clone\(\)](#)

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

Usage:

```
AtomAbstractObject$new(  
  xml = NULL,  
  element = NULL,  
  namespace = NULL,  
  attrs = list(),  
  defaults = list(),  
  wrap = TRUE,  
  logger = "INFO"  
)
```

Arguments:

xml object of class [XMLInternalNode-class](#)
element element
namespace namespace
attrs attrs
defaults defaults
wrap wrap
logger logger type

Method `setIsDocument()`: Set if object is a document or not

Usage:

```
AtomAbstractObject$setIsDocument(isDocument)
```

Arguments:

isDocument object of class logical

Method `isDocument()`: Informs if the object is a document

Usage:

```
AtomAbstractObject$isDocument()
```

Returns: object of class logical

Method `getRootElement()`: Get root XML element

Usage:

`AtomAbstractObject$getRootElement()`

Returns: object of class character

Method `getNamespace()`: Get XML namespace

Usage:

`AtomAbstractObject$getNamespace()`

Returns: object of class character

Method `createElement()`: Creates an element

Usage:

`AtomAbstractObject$createElement(element, type = "text")`

Arguments:

element element

type type. Default is "text"

Returns: the typed element

Method `addListElement()`: Add a metadata element to an element list

Usage:

`AtomAbstractObject$addListElement(field, metadataElement)`

Arguments:

field field

metadataElement metadata element to add

Returns: TRUE if added, FALSE otherwise

Method `dellistElement()`: Deletes a metadata element from an element list

Usage:

`AtomAbstractObject$dellistElement(field, metadataElement)`

Arguments:

field field

metadataElement metadata element to add

Returns: TRUE if deleted, FALSE otherwise

Method `contains()`: Indicates if an element list contains or not an element

Usage:

`AtomAbstractObject$contains(field, metadataElement)`

Arguments:

field field

metadataElement metadata element to add

Returns: TRUE if contained, FALSE otherwise

Method print(): Prints the element

Usage:

```
AtomAbstractObject#print(..., depth = 1)
```

Arguments:

... any parameter to pass to print method
depth printing depth

Method decode(): Decodes the object from an XML representation

Usage:

```
AtomAbstractObject$decode(xml)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from XML

Method encode(): Encodes the object as XML

Usage:

```
AtomAbstractObject$encode(  
  addNS = TRUE,  
  validate = TRUE,  
  strict = FALSE,  
  encoding = "UTF-8"  
)
```

Arguments:

addNS whether namespace has to be added. Default is TRUE
validate whether validation has to be done vs. XML schemas. Default is TRUE
strict whether strict validation has to be operated (raise an error if invalid). Default is FALSE
encoding encoding. Default is "UTF-8"

Method validate(): Validates the object / XML vs. XML schemas

Usage:

```
AtomAbstractObject$validate(xml = NULL, strict = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from XML
strict strict validation or not

Returns: TRUE if valid, FALSE otherwise

Method save(): Saves the object as XML file

Usage:

```
AtomAbstractObject$save(file, ...)
```

Arguments:

file file name
... any parameter to pass to encode() method

Method isFieldInheritedFrom(): Indicates the class from which field is inherited

Usage:

AtomAbstractObject\$isFieldInheritedFrom(field)

Arguments:

field field

Returns: an object of class [R6Class](#), or NULL

Method getClassname(): Get class name

Usage:

AtomAbstractObject\$getClassname()

Returns: object of class character

Method getClass(): Get class

Usage:

AtomAbstractObject\$class()

Returns: object of class [R6Class](#)

Method getNamespaceDefinition(): Get namespace definition

Usage:

AtomAbstractObject\$getNamespaceDefinition(recursive = FALSE)

Arguments:

recursive recursive

Returns: a named list of the XML namespaces

Method getXmlElement(): Get XML element name

Usage:

AtomAbstractObject\$xmlElement()

Returns: object of class character

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

Usage:

AtomAbstractObject\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Note

abstract class used internally by **atom4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

AtomAuthor	<i>Atom Author class</i>
------------	--------------------------

Description

This class models an Atom Author

Format

[R6Class](#) object.

Details

AtomAuthor

Value

Object of [R6Class](#) for modelling an Atom Author

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::AtomPerson](#) -> AtomAuthor

Methods

Public methods:

- [AtomAuthor\\$new\(\)](#)
- [AtomAuthor\\$clone\(\)](#)

Method `new()`: Initializes an [AtomAuthor](#)

Usage:

```
AtomAuthor$new(xml = NULL, name = NULL, uri = NULL, email = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

name name

uri uri

email email

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
  author <- AtomAuthor$new(name = "John Doe", email = "john.doe@atom4R.com")

## End(Not run)
```

AtomCategory

Atom Category class

Description

This class models an atom Category

Format

R6Class object.

Details

AtomCategory

Value

Object of R6Class for modelling an Atom Category

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> AtomCategory

Public fields

attrs attrs

value value

Methods**Public methods:**

- [AtomCategory\\$new\(\)](#)
- [AtomCategory\\$setTerm\(\)](#)
- [AtomCategory\\$setScheme\(\)](#)
- [AtomCategory\\$setLabel\(\)](#)
- [AtomCategory\\$clone\(\)](#)

Method new(): Initializes an [AtomCategory](#)

Usage:

```
AtomCategory$new(  
  xml = NULL,  
  value = NULL,  
  term = NULL,  
  scheme = NULL,  
  label = NULL  
)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value
term term
scheme scheme
label label

Method setTerm(): Set term

Usage:

```
AtomCategory$setTerm(term)
```

Arguments:

term term

Method setScheme(): Set scheme

Usage:

```
AtomCategory$setScheme(scheme)
```

Arguments:

scheme scheme

Method setLabel(): Set label

Usage:

```
AtomCategory$setLabel(label)
```

Arguments:

label label

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

AtomContributor	<i>Atom Contributor class</i>
-----------------	-------------------------------

Description

This class models an Atom Contributor

Format

[R6Class](#) object.

Details

AtomContributor

Value

Object of [R6Class](#) for modelling an Atom Contributor

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::AtomPerson](#) -> AtomContributor

Methods

Public methods:

- [AtomContributor\\$new\(\)](#)
- [AtomContributor\\$clone\(\)](#)

Method [new\(\)](#): Initializes an [AtomContributor](#)

Usage:

```
AtomContributor$new(xml = NULL, name = NULL, uri = NULL, email = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

name name

uri uri

email email

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:  
  contrib <- AtomContributor$new(name = "John Doe", email = "john.doe@atom4R.com")  
  
## End(Not run)
```

AtomEntry

Atom Entry class

Description

This class models an atom Entry

Format

[R6Class](#) object.

Details

AtomEntry

Value

Object of [R6Class](#) for modelling an Atom Entry

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> AtomEntry

Public fields

id identifier
updated Update date/time
published Publication date/time
title Title
summary Summary
rights Rights
source Source
author Author(s)
contributor Contributor(s)
category Category
content Content

Methods

Public methods:

- `AtomEntry$new()`
- `AtomEntry$setId()`
- `AtomEntry$setUpdated()`
- `AtomEntry$setPublished()`
- `AtomEntry$setTitle()`
- `AtomEntry$setSummary()`
- `AtomEntry$setRights()`
- `AtomEntry$setSource()`
- `AtomEntry$addAuthor()`
- `AtomEntry$delAuthor()`
- `AtomEntry$addContributor()`
- `AtomEntry$delContributor()`
- `AtomEntry$addCategory()`
- `AtomEntry$delCategory()`
- `AtomEntry$addLink()`
- `AtomEntry$delLink()`
- `AtomEntry$setContent()`
- `AtomEntry$clone()`

Method `new()`: Initializes an `AtomEntry`

Usage:

```
AtomEntry$new(xml = NULL)
```

Arguments:

`xml` object of class `XMLInternalNode-class` from `XML`

Method `setId()`: Set ID

Usage:

```
AtomEntry$setId(id)
```

Arguments:

`id` id

Method `setUpdated()`: Set updated date

Usage:

```
AtomEntry$setUpdated(updated)
```

Arguments:

`updated` object of class `Date` or `POSIXt`

Method `setPublished()`: Set published date

Usage:

```
AtomEntry$setPublished(published)
```


Arguments:

published object of class Date or POSIXt

Method setTitle(): Set title*Usage:*

```
AtomEntry$setTitle(title, type = "text")
```

Arguments:

title title

type type. Default is "text"

Method setSummary(): Set summary*Usage:*

```
AtomEntry$setSummary(summary, type = "text")
```

Arguments:

summary summary

type type. Default is "text"

Method setRights(): Set rights*Usage:*

```
AtomEntry$setRights(rights, type = "text")
```

Arguments:

rights rights

type type. Default is "text"

Method setSource(): Set source*Usage:*

```
AtomEntry$setSource(source, type = "text")
```

Arguments:

source source

type type. Default is "text"

Method addAuthor(): Adds author*Usage:*

```
AtomEntry$addAuthor(author)
```

Arguments:

author object of class [AtomAuthor](#)

Returns: TRUE if added, FALSE otherwise

Method delAuthor(): Deletes author*Usage:*

```
AtomEntry$delAuthor(author)
```

Arguments:

author object of class [AtomAuthor](#)

Returns: TRUE if deleted, FALSE otherwise

Method addContributor(): Adds contributor

Usage:

AtomEntry\$addContributor(contributor)

Arguments:

contributor object of class [AtomContributor](#)

Returns: TRUE if added, FALSE otherwise

Method delContributor(): Deletes contributor

Usage:

AtomEntry\$delContributor(contributor)

Arguments:

contributor object of class [AtomContributor](#)

Returns: TRUE if deleted, FALSE otherwise

Method addCategory(): Adds category

Usage:

AtomEntry\$addCategory(value, term, scheme = NULL, label = NULL)

Arguments:

value value

term term

scheme scheme

label label

Returns: TRUE if added, FALSE otherwise

Method delCategory(): Deletes category

Usage:

AtomEntry\$delCategory(value, term, scheme = NULL, label = NULL)

Arguments:

value value

term term

scheme scheme

label label

Returns: TRUE if deleted, FALSE otherwise

Method addLink(): Adds link

Usage:

AtomEntry\$addLink(link, rel = "alternate", type = "text/html")

Arguments:

link link
 rel relation. Default is "alternate"
 type type. Default is "text/html"
Returns: TRUE if added, FALSE otherwise

Method delLink(): Deletes link

Usage:
 AtomEntry\$delLink(link, rel = "alternate", type = "text/html")
Arguments:
 link link
 rel relation. Default is "alternate"
 type type. Default is "text/html"
Returns: TRUE if deleted, FALSE otherwise

Method setContent(): Set content

Usage:
 AtomEntry\$setContent(content)
Arguments:
 content content

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

Usage:
 AtomEntry\$clone(deep = FALSE)
Arguments:
 deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
## Not run:
#encoding
atom <- AtomEntry$new()
atom$setId("my-atom-entry")
atom$setTitle("My Atom feed entry")
atom$setSummary("My Atom feed entry very comprehensive abstract")
author1 <- AtomAuthor$new(
  name = "John Doe",
  uri = "http://www.atomxml.com/johndoe",
  email = "johndoe@atom4R.com"
)
atom$addAuthor(author1)
author2 <- AtomAuthor$new(
  name = "John Doe's sister",
```

```

    uri = "http://www.atomxml.com/johndoesister",
    email = "johndoesister@atom4R.com"
  )
  atom$addAuthor(author2)
  contrib1 <- AtomContributor$new(
    name = "Contrib1",
    uri = "http://www.atomxml.com/contrib1",
    email = "contrib1@atom4R.com"
  )
  atom$addContributor(contrib1)
  contrib2 <- AtomContributor$new(
    name = "Contrib2",
    uri = "http://www.atomxml.com/contrib2",
    email = "contrib2@atom4R.com"
  )
  atom$addContributor(contrib2)
  atom$addCategory("draft", "dataset")
  atom$addCategory("world", "spatial")
  atom$addCategory("fisheries", "domain")

  xml <- atom$encode()

## End(Not run)

```

AtomFeed

Atom feed class

Description

This class models an atom feed

Format

[R6Class](#) object.

Details

AtomFeed

Value

Object of [R6Class](#) for modelling an Atom feed

Methods

`new(xml)` This method is used to create an Atom Feed

`setId(id)` Set identifier

`setUpdated(updated)` Set update date (object of class 'character' or 'POSIX')

`addLink(link, rel, type)` Adds a link. Default rel value is set to "alternate". Default type value is set to "text/html"

`delLink(link, rel, type)` Deletes a link

`setSelfLink(link)` Sets a self-relation link

`setAlternateLink(link, type)` Sets an alternate-relation link. Default type is "text/html"

`setTitle(title)` Set title

`setSubtitle(subtitle)` Set subtitle

`addAuthor(author)` Adds an author, object of class `AtomAuthor`

`delAuthor(author)` Deletes an author, object of class `AtomAuthor`

`addContributor(contributor)` Adds a contributor, object of class `AtomContributor`

`delContributor(contributor)` Deletes a contributor, object of class `AtomContributor`

`setGenerator(generator, type)` Sets generator

`setIcon(icon)` Sets icon

`addCategory(term, scheme, label)` Adds a category

`delCategory(term, scheme, label)` Deletes a category

`addEntry(entry)` Adds an entry, object of class `AtomEntry`

`delEntry(entry)` Deletes an entry, object of class `AtomEntry`

Super classes

`atom4R::atom4RLogger` -> `atom4R::AtomAbstractObject` -> `AtomFeed`

Public fields

`id` Identifier

`updated` Update date

`published` Publication date

`title` Title

`subtitle` Subtitle

`rights` Rights (license, use, ...)

`author` Author person

`contributor` Contributor person

`generator` Generator

`icon` Icon

`logo` Logo

`category` Category

`link` links

`entry` List of entries

Methods

Public methods:

- [AtomFeed\\$new\(\)](#)
- [AtomFeed\\$setId\(\)](#)
- [AtomFeed\\$setUpdated\(\)](#)
- [AtomFeed\\$setPublished\(\)](#)
- [AtomFeed\\$addLink\(\)](#)
- [AtomFeed\\$delLink\(\)](#)
- [AtomFeed\\$setSelfLink\(\)](#)
- [AtomFeed\\$setAlternateLink\(\)](#)
- [AtomFeed\\$setTitle\(\)](#)
- [AtomFeed\\$setSubtitle\(\)](#)
- [AtomFeed\\$setRights\(\)](#)
- [AtomFeed\\$addAuthor\(\)](#)
- [AtomFeed\\$delAuthor\(\)](#)
- [AtomFeed\\$addContributor\(\)](#)
- [AtomFeed\\$delContributor\(\)](#)
- [AtomFeed\\$setGenerator\(\)](#)
- [AtomFeed\\$setIcon\(\)](#)
- [AtomFeed\\$addCategory\(\)](#)
- [AtomFeed\\$delCategory\(\)](#)
- [AtomFeed\\$addEntry\(\)](#)
- [AtomFeed\\$delEntry\(\)](#)
- [AtomFeed\\$clone\(\)](#)

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

Usage:

```
AtomFeed$new(xml = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

Method [setId\(\)](#): Set ID

Usage:

```
AtomFeed$setId(id)
```

Arguments:

id id

Method [setUpdated\(\)](#): Set updated date

Usage:

```
AtomFeed$setUpdated(updated)
```

Arguments:

updated object of class [Date](#) or [POSIXt](#)

Method setPublished(): Set published date

Usage:

```
AtomFeed$setPublished(published)
```

Arguments:

published object of class Date or POSIXt

Method addLink(): Adds link

Usage:

```
AtomFeed$addLink(link, rel = "alternate", type = "text/html")
```

Arguments:

link link

rel relation. Default is "alternate"

type type. Default is "text/html"

Returns: TRUE if added, FALSE otherwise

Method delLink(): Deletes link

Usage:

```
AtomFeed$delLink(link, rel = "alternate", type = "text/html")
```

Arguments:

link link

rel relation. Default is "alternate"

type type. Default is "text/html"

Returns: TRUE if deleted, FALSE otherwise

Method setSelfLink(): Set self link

Usage:

```
AtomFeed$setSelfLink(link)
```

Arguments:

link link

Returns: TRUE if set, FALSE otherwise

Method setAlternateLink(): Set alternate link

Usage:

```
AtomFeed$setAlternateLink(link, type = "text/html")
```

Arguments:

link link

type type. Default is "text/html"

Returns: TRUE if set, FALSE otherwise

Method setTitle(): Set title

Usage:

AtomFeed\$setTitle(title, type = "text")

Arguments:

title title

type type. Default is "text"

Method setSubtitle(): Set subtitle

Usage:

AtomFeed\$setSubtitle(subtitle, type = "text")

Arguments:

subtitle subtitle

type type. Default is "text"

Method setRights(): Set rights

Usage:

AtomFeed\$setRights(rights, type = "text")

Arguments:

rights rights

type type. Default is "text"

Method addAuthor(): Adds author

Usage:

AtomFeed\$addAuthor(author)

Arguments:

author object of class [AtomAuthor](#)

Returns: TRUE if added, FALSE otherwise

Method delAuthor(): Deletes author

Usage:

AtomFeed\$delAuthor(author)

Arguments:

author object of class [AtomAuthor](#)

Returns: TRUE if deleted, FALSE otherwise

Method addContributor(): Adds contributor

Usage:

AtomFeed\$addContributor(contributor)

Arguments:

contributor object of class [AtomContributor](#)

Returns: TRUE if added, FALSE otherwise

Method delContributor(): Deletes contributor

Usage:

AtomFeed\$delContributor(contributor)

Arguments:

contributor object of class [AtomContributor](#)

Returns: TRUE if deleted, FALSE otherwise

Method setGenerator(): Set generator

Usage:

AtomFeed\$setGenerator(generator, type = "text")

Arguments:

generator generator

type type. Default is "text"

Method setIcon(): Set icon

Usage:

AtomFeed\$setIcon(icon)

Arguments:

icon icon

Method addCategory(): Adds category

Usage:

AtomFeed\$addCategory(value, term, scheme = NULL, label = NULL)

Arguments:

value value

term term

scheme scheme

label label

Returns: TRUE if added, FALSE otherwise

Method delCategory(): Deletes category

Usage:

AtomFeed\$delCategory(value, term, scheme = NULL, label = NULL)

Arguments:

value value

term term

scheme scheme

label label

Returns: TRUE if deleted, FALSE otherwise

Method addEntry(): Adds an entry

Usage:

AtomFeed\$addEntry(entry)

Arguments:

entry object of class [AtomEntry](#)

Returns: TRUE if added, FALSE otherwise

Method delEntry(): Deletes an entry

Usage:

```
AtomFeed$delEntry(entry)
```

Arguments:

entry object of class [AtomEntry](#)

Returns: TRUE if deleted, FALSE otherwise

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
#encoding
atom <- AtomFeed$new()
atom$setId("my-atom-feed")
atom$setTitle("My Atom feed title")
atom$setSubtitle("MyAtom feed subtitle")
author1 <- AtomAuthor$new(
  name = "John Doe",
  uri = "http://www.atomxml.com/johndoe",
  email = "johndoe@atom4R.com"
)
atom$addAuthor(author1)
author2 <- AtomAuthor$new(
  name = "John Doe's sister",
  uri = "http://www.atomxml.com/johndoesister",
  email = "johndoesister@atom4R.com"
)
atom$addAuthor(author2)
contrib1 <- AtomContributor$new(
  name = "Contrib1",
  uri = "http://www.atomxml.com/contrib1",
  email = "contrib1@atom4R.com"
)
atom$addContributor(contrib1)
contrib2 <- AtomContributor$new(
  name = "Contrib2",
  uri = "http://www.atomxml.com/contrib2",
  email = "contrib2@atom4R.com"
```

```

)
atom$addContributor(contrib2)
atom$setIcon("https://via.placeholder.com/300x150.png/03f/fff?text=atom4R")
atom$selfLink("http://example.com/atom.feed")
atom$setAlternateLink("http://example.com/my-atom-feed")
atom$addCategory("draft", "dataset")
atom$addCategory("world", "spatial")
atom$addCategory("fisheries", "domain")
#add entry
entry <- AtomEntry$new()
entry$setId("my-atom-entry")
entry$setTitle("My Atom feed entry")
entry$setSummary("My Atom feed entry very comprehensive abstract")
author1 <- AtomAuthor$new(
  name = "John Doe",
  uri = "http://www.atomxml.com/johndoe",
  email = "johndoe@atom4R.com"
)
entry$addAuthor(author1)
author2 <- AtomAuthor$new(
  name = "John Doe's sister",
  uri = "http://www.atomxml.com/johndoesister",
  email = "johndoesister@atom4R.com"
)
entry$addAuthor(author2)
contrib1 <- AtomContributor$new(
  name = "Contrib1",
  uri = "http://www.atomxml.com/contrib1",
  email = "contrib1@atom4R.com"
)
entry$addContributor(contrib1)
contrib2 <- AtomContributor$new(
  name = "Contrib2",
  uri = "http://www.atomxml.com/contrib2",
  email = "contrib2@atom4R.com"
)
entry$addContributor(contrib2)
entry$addCategory("draft", "dataset")
entry$addCategory("world", "spatial")
entry$addCategory("fisheries", "domain")
atom$addEntry(entry)
xml <- atom$encode()

```

AtomLink

Atom Link class

Description

This class models an atom Link

Format

[R6Class](#) object.

Details

AtomLink

Value

Object of [R6Class](#) for modelling an Atom Link

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> AtomLink

Public fields

attrs attrs

Methods**Public methods:**

- [AtomLink\\$new\(\)](#)
- [AtomLink\\$setRel\(\)](#)
- [AtomLink\\$setType\(\)](#)
- [AtomLink\\$setHref\(\)](#)
- [AtomLink\\$setHreflang\(\)](#)
- [AtomLink\\$setTitle\(\)](#)
- [AtomLink\\$setLength\(\)](#)
- [AtomLink\\$clone\(\)](#)

Method `new()`: Initializes an [AtomLink](#)

Usage:

```
AtomLink$new(  
  xml = NULL,  
  rel = NULL,  
  type = NULL,  
  href = NULL,  
  hreflang = NULL,  
  title = NULL,  
  length = NULL  
)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
rel rel
type type

href href
hreflang hreflang
title title
length length

Method setRel(): Set relation

Usage:

AtomLink\$setRel(rel)

Arguments:

rel rel

Method setType(): Set type

Usage:

AtomLink\$setType(type)

Arguments:

type type

Method setHref(): Set href

Usage:

AtomLink\$setHref(href)

Arguments:

href href

Method setHreflang(): Set href lang

Usage:

AtomLink\$setHreflang(hreflang)

Arguments:

hreflang hreflang

Method setTitle(): Set title

Usage:

AtomLink\$setTitle(title)

Arguments:

title title

Method setLength(): Set length

Usage:

AtomLink\$setLength(length)

Arguments:

length length

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

Usage:

AtomLink\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

AtomNamespace

AtomNamespace

Description

AtomNamespace

AtomNamespace

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Atom Namespace

Public fields

id id

uri uri

Methods**Public methods:**

- [AtomNamespace\\$new\(\)](#)
- [AtomNamespace\\$getDefinition\(\)](#)
- [AtomNamespace\\$clone\(\)](#)

Method [new\(\)](#): Initializes an [AtomNamespace](#)

Usage:

`AtomNamespace$new(id, uri)`

Arguments:

id id

uri uri

Method [getDefinition\(\)](#): Get definition

Usage:

`AtomNamespace$getDefinition()`

Returns: a named list defining the namespace

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

Usage:

`AtomNamespace$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

Note

ISO class used internally by atom4R for specifying XML namespaces

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

AtomPerson

Atom Person class

Description

This class models an Atom Person

Format

[R6Class](#) object.

Details

AtomPerson

Value

Object of [R6Class](#) for modelling an Atom Person

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> AtomPerson

Public fields

name name

uri uri

email email

Methods**Public methods:**

- [AtomPerson\\$new\(\)](#)
- [AtomPerson\\$setName\(\)](#)
- [AtomPerson\\$setUri\(\)](#)
- [AtomPerson\\$setEmail\(\)](#)
- [AtomPerson\\$clone\(\)](#)

Method [new\(\)](#): Initializes an [AtomPerson](#)

Usage:

```
AtomPerson$new(xml = NULL, name = NULL, uri = NULL, email = NULL)
```

Arguments:

```
xml  object of class XMLInternalNode-class from XML  
name name  
uri  uri  
email email
```

Method setName(): Set name*Usage:*

```
AtomPerson$setName(name)
```

Arguments:

```
name name
```

Method setUri(): Set URI*Usage:*

```
AtomPerson$setUri(uri)
```

Arguments:

```
uri uri
```

Method setEmail(): Set email*Usage:*

```
AtomPerson$setEmail(email)
```

Arguments:

```
email email
```

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

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

Arguments:

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

Note

Abstract class used internally for person-like classes

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

AtomPubClient	<i>AtomPubClient class</i>
---------------	----------------------------

Description

This class models an AtomPub service client

Format

[R6Class](#) object.

Details

AtomPubClient

Value

Object of [R6Class](#) for modelling an AtomPub client

Methods

`new(url, user, pwd, token, keyring_backend)` This method is to instantiate an AtomPub Client.

The `keyring_backend` can be set to use a different backend for storing the Atom pub user token with **keyring** (Default value is 'env').

The logger can be either NULL, "INFO" (with minimum logs), or "DEBUG" (for complete curl http calls logs)

`getUser()` Retrieves user (if any specified).

`getPwd()` Retrieves user password (if any user specified).

`getToken()` Retrieves user token.

`getServiceDocument()` Gets service document description. Unimplemented in abstract classes.

`listCollections(pretty)` Lists the available collections. Use `pretty` to return a "data.frame" instead of a list.

`getCollectionMembers(collectionId)` List members of a collection. Unimplemented in abstract classes.

Super class

`atom4R::atom4RLogger` -> AtomPubClient

Public fields

service service

Methods

Public methods:

- [AtomPubClient\\$new\(\)](#)
- [AtomPubClient\\$getUser\(\)](#)
- [AtomPubClient\\$getPwd\(\)](#)
- [AtomPubClient\\$getToken\(\)](#)
- [AtomPubClient\\$getServiceDocument\(\)](#)
- [AtomPubClient\\$listCollections\(\)](#)
- [AtomPubClient\\$getCollectionMembers\(\)](#)
- [AtomPubClient\\$clone\(\)](#)

Method `new()`: This method is to instantiate an SWORD Client. By default the version is set to "2".

The `keyring_backend` can be set to use a different backend for storing the SWORD API user token with **keyring** (Default value is 'env').

The logger allows to specify the level of log (default is NULL), either "INFO" for **atom4R** logs or "DEBUG" for verbose HTTP client (curl) logs.

Usage:

```
AtomPubClient$new(
  url,
  user = NULL,
  pwd = NULL,
  token = NULL,
  logger = NULL,
  keyring_backend = "env"
)
```

Arguments:

```
url url
user user
pwd pwd
token token
logger logger
keyring_backend keyring backend. Default is 'env'
```

Method `getUser()`: Get user

Usage:

```
AtomPubClient$getUser()
```

Returns: object of class character

Method `getPwd()`: Get password

Usage:

```
AtomPubClient$getPwd()
```

Returns: object of class character

Method getToken(): Get token

Usage:

AtomPubClient\$getToken()

Returns: object of class character

Method getServiceDocument(): Get service document

Usage:

AtomPubClient\$getServiceDocument()

Arguments:

force force Force getting/refreshing of service document

Returns: object of class [SwordServiceDocument](#)

Method listCollections(): List collections

Usage:

AtomPubClient\$listCollections(pretty = FALSE)

Arguments:

pretty pretty

Returns: a list of collections, or data.frame

Method getCollectionMembers(): Get collection members. Unimplemented abstract method at [AtomPubClient](#) level

Usage:

AtomPubClient\$getCollectionMembers()

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

Usage:

AtomPubClient\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Note

Abstract class used internally for AtomPub (Atom Publishing Protocol) clients

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

DCAbstract

DCAbstract

Description

This class models an DublinCore 'abstract' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'abstract' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDescription](#)
-> [DCAbstract](#)

Methods

Public methods:

- [DCAbstract\\$new\(\)](#)
- [DCAbstract\\$clone\(\)](#)

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

Usage:

[DCAbstract\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

[DCAbstract\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/abstract>

DCAccessRights	<i>DCAccessRights</i>
----------------	-----------------------

Description

This class models an DublinCore 'accessRights' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'accessRights' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRights](#)
-> [DCAccessRights](#)

Methods

Public methods:

- [DCAccessRights\\$new\(\)](#)
- [DCAccessRights\\$clone\(\)](#)

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

Usage:

```
DCAccessRights$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/accessRights>

DCAccrualMethod

DCAccrualMethod

Description

This class models an DublinCore 'accrualMethod' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'accrualMethod' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCAccrualMethod

Methods

Public methods:

- [DCAccrualMethod\\$new\(\)](#)
- [DCAccrualMethod\\$clone\(\)](#)

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

Usage:

```
DCAccrualMethod$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/accrualMethod>

DCAccrualPeriodicity *DCAccrualPeriodicity*

Description

This class models an DublinCore 'accrualPeriodicity' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'accrualPeriodicity' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [DCAccrualPeriodicity](#)

Methods**Public methods:**

- [DCAccrualPeriodicity\\$new\(\)](#)
- [DCAccrualPeriodicity\\$clone\(\)](#)

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

Usage:

```
DCAccrualPeriodicity$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/accrualPeriodicity>

DCAccrualPolicy

DCAccrualPolicy

Description

This class models an DublinCore 'accrualPolicy' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'accrualPolicy' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [DCAccrualPolicy](#)

Methods

Public methods:

- [DCAccrualPolicy\\$new\(\)](#)
- [DCAccrualPolicy\\$clone\(\)](#)

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

Usage:

```
DCAccrualPolicy$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/accrualPolicy>

DCAlternative

DCAlternative

Description

This class models an DublinCore 'alternative' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'alternative' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCTitle](#)
-> [DCAlternative](#)

Methods

Public methods:

- [DCAlternative\\$new\(\)](#)
- [DCAlternative\\$clone\(\)](#)

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

Usage:

[DCAlternative\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

[DCAlternative\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/alternative>

DCAudience

DCAudience

Description

This class models an DublinCore 'audience' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'audience' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCAudience

Methods

Public methods:

- [DCAudience\\$new\(\)](#)
- [DCAudience\\$clone\(\)](#)

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

Usage:

```
DCAudience$new(xml = NULL, term = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/audience>

DCAvailable

DCAvailable

Description

This class models an DublinCore 'available' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'available' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCAvailable](#)

Methods

Public methods:

- [DCAvailable\\$new\(\)](#)
- [DCAvailable\\$clone\(\)](#)

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

Usage:

```
DCAvailable$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/available>

DCBibliographicCitation

DCBibliographicCitation

Description

This class models an DublinCore 'bibliographicCitation' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'bibliographicCitation' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCIdentifier](#)
-> [DCBibliographicCitation](#)

Methods

Public methods:

- [DCBibliographicCitation\\$new\(\)](#)
- [DCBibliographicCitation\\$clone\(\)](#)

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

Usage:

[DCBibliographicCitation\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

value value

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

Usage:

[DCBibliographicCitation\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/bibliographicCitation/>

DCConformsTo

DCConformsTo

Description

This class models an DublinCore 'conformsTo' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'conformsTo' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRelation](#)
-> DCConformsTo

Methods

Public methods:

- [DCConformsTo\\$new\(\)](#)
- [DCConformsTo\\$clone\(\)](#)

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

Usage:

`DCConformsTo$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

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

Usage:

`DCConformsTo$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/conformsTo>

DCContributor

DCContributor

Description

This class models an DublinCore 'contributor' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'contributor' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCContributor

Methods

Public methods:

- [DCContributor\\$new\(\)](#)
- [DCContributor\\$clone\(\)](#)

Method `new()`: This method is used to create an Dublin core 'contributor' element. Use `dc` to `TRUE` to use Dublin core namespace instead of DC terms.

Usage:

```
DCContributor$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

`dc` use DC namespace?

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/contributor>

DCCoverage

DCCoverage

Description

This class models an DublinCore Terms 'coverage' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'coverage' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCCoverage

Methods

Public methods:

- [DCCoverage\\$new\(\)](#)
- [DCCoverage\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'coverage' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCCoverage$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
term term
value value
dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/coverage>

DCCreated

DCCreated

Description

This class models an DublinCore Terms 'date' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'date' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> DCCreated

Methods

Public methods:

- [DCCreated\\$new\(\)](#)
- [DCCreated\\$clone\(\)](#)

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

Usage:

`DCCreated$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from [XML](#)
`value` value

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

Usage:

`DCCreated$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/created>

DCCreator

DCCreator

Description

This class models an DublinCore 'creator' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'creator' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCCreator

Methods

Public methods:

- [DCCreator\\$new\(\)](#)
- [DCCreator\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'creator' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCCreator$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/creator>

DCDate

DCDate

Description

This class models an DublinCore 'date' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'date' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCDate

Methods

Public methods:

- [DCDate\\$new\(\)](#)
- [DCDate\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'date' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCDate$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/date>

DCDateAccepted	<i>DCDateAccepted</i>
----------------	-----------------------

Description

This class models an DublinCore 'dateAccepted' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'dateAccepted' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCDateAccepted](#)

Methods

Public methods:

- [DCDateAccepted\\$new\(\)](#)
- [DCDateAccepted\\$clone\(\)](#)

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

Usage:

```
DCDateAccepted$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/dateAccepted>

DCDateCopyrighted *DCDateCopyrighted*

Description

This class models an DublinCore 'dateCopyrighted' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'dateCopyrighted' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCDateCopyrighted](#)

Methods

Public methods:

- [DCDateCopyrighted\\$new\(\)](#)
- [DCDateCopyrighted\\$clone\(\)](#)

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

Usage:

```
DCDateCopyrighted$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/dateCopyrighted>

DCDateSubmitted	<i>DCDateSubmitted</i>
-----------------	------------------------

Description

This class models an DublinCore 'dateSubmitted' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'dateSubmitted' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCDateSubmitted](#)

Methods

Public methods:

- [DCDateSubmitted\\$new\(\)](#)
- [DCDateSubmitted\\$clone\(\)](#)

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

Usage:

[DCDateSubmitted\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
value value

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

Usage:

[DCDateSubmitted\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/dateSubmitted>

DCDescription

DCDescription

Description

This class models an DublinCore 'description' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'description' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCDescription

Methods**Public methods:**

- [DCDescription\\$new\(\)](#)
- [DCDescription\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'description' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCDescription$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/description>

DCEducationalLevel *DCEducationalLevel*

Description

This class models an DublinCore 'educationalLevel' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'educationalLevel' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCAudience](#)
-> [DCEducationalLevel](#)

Methods

Public methods:

- [DCEducationalLevel\\$new\(\)](#)
- [DCEducationalLevel\\$clone\(\)](#)

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

Usage:

[DCEducationalLevel\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

[DCEducationalLevel\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/educationalLevel>

DCElement

DublinCore element class

Description

This class models an DublinCore element

Format

[R6Class](#) object.

Details

DCElement

Value

Object of [R6Class](#) for modelling an Dublin Core element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> DCElement

Public fields

value value

Methods

Public methods:

- [DCElement\\$new\(\)](#)
- [DCElement\\$clone\(\)](#)

Method `new()`: Initializes an abstract [DCElement](#)

Usage:

```
DCElement$new(  
  xml = NULL,  
  term = NULL,  
  value = NULL,  
  vocabulary = NULL,  
  extended = FALSE  
)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
term term
value value

vocabulary vocabulary
extended extended

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

Usage:

DCElement\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Note

Class used internally by **atom4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

DCEntry

Dublin Core Entry class

Description

This class models an Dublin Core Entry

Format

[R6Class](#) object.

Details

DCEntry

Value

Object of [R6Class](#) for modelling an Dublin Core Entry

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::AtomEntry](#) -> DCEntry

Methods**Public methods:**

- `DCEntry$new()`
- `DCEntry$addDCElement()`
- `DCEntry$delDCElement()`
- `DCEntry$setDCElements()`
- `DCEntry$getDCElements()`
- `DCEntry$getElementByValue()`
- `DCEntry$addDCAbstract()`
- `DCEntry$delDCAbstract()`
- `DCEntry$setDCAbstracts()`
- `DCEntry$getDCAbstracts()`
- `DCEntry$addDCAccessRights()`
- `DCEntry$delDCAccessRights()`
- `DCEntry$setDCAccessRights()`
- `DCEntry$getDCAccessRights()`
- `DCEntry$addDCAccrualMethod()`
- `DCEntry$delDCAccrualMethod()`
- `DCEntry$setDCAccrualMethods()`
- `DCEntry$getDCAccrualMethods()`
- `DCEntry$addDCAccrualPeriodicity()`
- `DCEntry$delDCAccrualPeriodicity()`
- `DCEntry$setDCAccrualPeriodicities()`
- `DCEntry$getDCAccrualPeriodicities()`
- `DCEntry$addDCAccrualPolicy()`
- `DCEntry$delDCAccrualPolicy()`
- `DCEntry$setDCAccrualPolicies()`
- `DCEntry$getDCAccrualPolicies()`
- `DCEntry$addDCAAlternative()`
- `DCEntry$delDCAAlternative()`
- `DCEntry$setDCAAlternatives()`
- `DCEntry$getDCAAlternatives()`
- `DCEntry$addDCAudience()`
- `DCEntry$delDCAudience()`
- `DCEntry$setDCAudiences()`
- `DCEntry$getDCAudiences()`
- `DCEntry$addDCAvailable()`
- `DCEntry$delDCAvailable()`
- `DCEntry$setDCAvailables()`
- `DCEntry$getDCAvailables()`
- `DCEntry$addDCBibliographicCitation()`
- `DCEntry$delDCBibliographicCitation()`

- DCEntry\$setDCBibliographicCitations()
- DCEntry\$getDCBibliographicCitations()
- DCEntry\$addDCConformsTo()
- DCEntry\$delDCConformsTo()
- DCEntry\$setDCConformsTo()
- DCEntry\$getDCConformsTo()
- DCEntry\$addDCContributor()
- DCEntry\$delDCContributor()
- DCEntry\$setDCContributors()
- DCEntry\$getDCContributors()
- DCEntry\$addDCCoverage()
- DCEntry\$delDCCoverage()
- DCEntry\$setDCCoverages()
- DCEntry\$getDCCoverages()
- DCEntry\$addDCCreated()
- DCEntry\$delDCCreated()
- DCEntry\$addDCCreator()
- DCEntry\$delDCCreator()
- DCEntry\$setDCCreators()
- DCEntry\$getDCCreators()
- DCEntry\$addDCDate()
- DCEntry\$delDCDate()
- DCEntry\$setDCDates()
- DCEntry\$getDCDates()
- DCEntry\$addDCDateAccepted()
- DCEntry\$delDCDateAccepted()
- DCEntry\$addDCDateCopyrighted()
- DCEntry\$delDCDateCopyrighted()
- DCEntry\$addDCDateSubmitted()
- DCEntry\$delDCDateSubmitted()
- DCEntry\$addDCDescription()
- DCEntry\$delDCDescription()
- DCEntry\$setDCDescriptions()
- DCEntry\$getDCDescriptions()
- DCEntry\$addDCEducationalLevel()
- DCEntry\$delDCEducationalLevel()
- DCEntry\$setDCEducationalLevels()
- DCEntry\$getDCEducationalLevels()
- DCEntry\$addDCExtent()
- DCEntry\$delDCExtent()
- DCEntry\$setDCExtents()
- DCEntry\$getDCExtents()

- DCEntry\$addDCFormat()
- DCEntry\$delDCFormat()
- DCEntry\$setDCFormats()
- DCEntry\$getDCFormats()
- DCEntry\$addDCHasPart()
- DCEntry\$delDCHasPart()
- DCEntry\$setDCHasParts()
- DCEntry\$getDCHasParts()
- DCEntry\$addDCHasVersion()
- DCEntry\$delDCHasVersion()
- DCEntry\$setDCHasVersions()
- DCEntry\$getDCHasVersions()
- DCEntry\$addDCIdentifier()
- DCEntry\$delDCIdentifier()
- DCEntry\$setDCIdentifiers()
- DCEntry\$getDCIdentifiers()
- DCEntry\$addDCInstructionalMethod()
- DCEntry\$delDCInstructionalMethod()
- DCEntry\$setDCInstructionalMethods()
- DCEntry\$getDCInstructionalMethods()
- DCEntry\$addDCIsPartOf()
- DCEntry\$delDCIsPartOf()
- DCEntry\$setDCIsPartOf()
- DCEntry\$getDCIsPartOfs()
- DCEntry\$addDCIsReferencedBy()
- DCEntry\$delDCIsReferencedBy()
- DCEntry\$setDCIsReferencedBy()
- DCEntry\$getDCIsReferencedBy()
- DCEntry\$addDCIsReplacedBy()
- DCEntry\$delDCIsReplacedBy()
- DCEntry\$setDCIsReplacedBy()
- DCEntry\$getDCIsReplacedBy()
- DCEntry\$addDCIsRequiredBy()
- DCEntry\$delDCIsRequiredBy()
- DCEntry\$setDCIsRequiredBy()
- DCEntry\$getDCIsRequiredBy()
- DCEntry\$addDCIsVersionOf()
- DCEntry\$delDCIsVersionOf()
- DCEntry\$setDCIsVersionOfs()
- DCEntry\$getDCIsVersionOfs()
- DCEntry\$addDCIssued()
- DCEntry\$delDCIssued()

- `DCEntity$addDCLanguage()`
- `DCEntity$delDCLanguage()`
- `DCEntity$setDCLanguages()`
- `DCEntity$getDCLanguages()`
- `DCEntity$addDCLicense()`
- `DCEntity$delDCLicense()`
- `DCEntity$setDCLicenses()`
- `DCEntity$getDCLicenses()`
- `DCEntity$addDCMediator()`
- `DCEntity$delDCMediator()`
- `DCEntity$setDCMediators()`
- `DCEntity$getDCMediators()`
- `DCEntity$addDCMedium()`
- `DCEntity$delDCMedium()`
- `DCEntity$setDCMediums()`
- `DCEntity$getDCMediums()`
- `DCEntity$addDCModified()`
- `DCEntity$delDCModified()`
- `DCEntity$addDCProvenance()`
- `DCEntity$delDCProvenance()`
- `DCEntity$setDCProvenances()`
- `DCEntity$getDCProvenances()`
- `DCEntity$addDCPublisher()`
- `DCEntity$delDCPublisher()`
- `DCEntity$setDCPublishers()`
- `DCEntity$getDCPublishers()`
- `DCEntity$addDCReferences()`
- `DCEntity$delDCReferences()`
- `DCEntity$setDCReferences()`
- `DCEntity$getDCReferences()`
- `DCEntity$addDCRelation()`
- `DCEntity$delDCRelation()`
- `DCEntity$setDCRelations()`
- `DCEntity$getDCRelations()`
- `DCEntity$addDCReplaces()`
- `DCEntity$delDCReplaces()`
- `DCEntity$setDCReplaces()`
- `DCEntity$getDCReplaces()`
- `DCEntity$addDCRequires()`
- `DCEntity$delDCRequires()`
- `DCEntity$setDCRequires()`
- `DCEntity$getDCRequires()`

- `DCEntry$addDCRights()`
- `DCEntry$delDCRights()`
- `DCEntry$setDCRights()`
- `DCEntry$getDCRights()`
- `DCEntry$addDCRightsHolder()`
- `DCEntry$delDCRightsHolder()`
- `DCEntry$setDCRightsHolders()`
- `DCEntry$getDCRightsHolders()`
- `DCEntry$addDCSource()`
- `DCEntry$delDCSource()`
- `DCEntry$setDCSources()`
- `DCEntry$getDCSources()`
- `DCEntry$addDCSubject()`
- `DCEntry$delDCSubject()`
- `DCEntry$setDCSubjects()`
- `DCEntry$getDCSubjects()`
- `DCEntry$addDCTableOfContents()`
- `DCEntry$delDCTableOfContents()`
- `DCEntry$setDCTablesOfContents()`
- `DCEntry$getDCTablesOfContent()`
- `DCEntry$addDCTemporal()`
- `DCEntry$delDCTemporal()`
- `DCEntry$setDCTemporals()`
- `DCEntry$getDCTemporals()`
- `DCEntry$addDCTitle()`
- `DCEntry$delDCTitle()`
- `DCEntry$setDCTitles()`
- `DCEntry$getDCTitles()`
- `DCEntry$addDCType()`
- `DCEntry$delDCType()`
- `DCEntry$setDCTypes()`
- `DCEntry$getDCTypes()`
- `DCEntry$clone()`

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

Usage:

```
DCEntry$new(xml = NULL)
```

Arguments:

`xml` object of class `XMLInternalNode-class` from `XML`

Method `addDCElement()`: Adds a Dublin Core element

Usage:

DCEntry\$addDCElement(term, value, extended = FALSE)

Arguments:

term term

value value

extended extended. Default is FALSE

Returns: TRUE if added, FALSE otherwise

Method delDCElement(): Deletes a Dublin Core element

Usage:

DCEntry\$delDCElement(term, value)

Arguments:

term term

value value

Returns: TRUE if deleted, FALSE otherwise

Method setDCElements(): Set a list of DC elements

Usage:

DCEntry\$setDCElements(term, values)

Arguments:

term term

values vector of values

Method getDCElements(): Get a list of DC elements

Usage:

DCEntry\$getDCElements(term)

Arguments:

term term

Returns: a list of objects extending [DCElement](#)

Method getDCElementByValue(): Get a DC element by value

Usage:

DCEntry\$getDCElementByValue(term, value)

Arguments:

term term

value value

Method addDCAbstract(): Adds DC abstract

Usage:

DCEntry\$addDCAbstract(abstract)

Arguments:

abstract object of class [DCAbstract](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAbstract(): Deletes DC abstract

Usage:

DCEnter\$delDCAbstract(abstract)

Arguments:

abstract object of class [DCAbstract](#) or vector of class [character](#) and length 1

Method setDCAbstracts(): Set DC abstracts

Usage:

DCEnter\$setDCAbstracts(abstracts)

Arguments:

abstracts abstracts, vector of class [character](#)

Method getDCAbstracts(): Get DC abstracts

Usage:

DCEnter\$getDCAbstracts()

Returns: a list of objects of class [DCAbstract](#)

Method addDCAccessRights(): Adds DC access rights

Usage:

DCEnter\$addDCAccessRights(accessRights)

Arguments:

accessRights object of class [DCAccessRights](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAccessRights(): Deletes DC access rights

Usage:

DCEnter\$delDCAccessRights(accessRights)

Arguments:

accessRights object of class [DCAccessRights](#) or vector of class [character](#) and length 1

Method setDCAccessRights(): Set access rights

Usage:

DCEnter\$setDCAccessRights(accessRights)

Arguments:

accessRights vector of class [character](#)

Method getDCAccessRights(): Get DC access rights

Usage:

DCEnter\$getDCAccessRights()

Returns: a list of objects of class [DCAccessRights](#)

Method addDCAccrualMethod(): Adds DC accrual method

Usage:

DCEntry\$addDCAccrualMethod(accrualMethod)

Arguments:

accrualMethod object of class [DCAccrualMethod](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAccrualMethod(): Deletes DC accrual method

Usage:

DCEntry\$delDCAccrualMethod(accrualMethod)

Arguments:

accrualMethod object of class [DCAccrualMethod](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAccrualMethods(): Set DC accrual method

Usage:

DCEntry\$setDCAccrualMethods(accrualMethods)

Arguments:

accrualMethods vector of class [character](#)

Method getDCAccrualMethods(): Get DC accrual method

Usage:

DCEntry\$getDCAccrualMethods()

Returns: a list of objects of class [DCAccrualMethod](#)

Method addDCAccrualPeriodicity(): Adds DC accrual periodicity

Usage:

DCEntry\$addDCAccrualPeriodicity(accrualPeriodicity)

Arguments:

accrualPeriodicity object of class [DCAccrualPeriodicity](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAccrualPeriodicity(): Deletes DC accrual periodicity

Usage:

DCEntry\$delDCAccrualPeriodicity(accrualPeriodicity)

Arguments:

accrualPeriodicity object of class [DCAccrualPeriodicity](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAccrualPeriodicities(): Set DC accrual periodicities

Usage:

DCEntry\$setDCAccrualPeriodicities(accrualPeriodicities)

Arguments:

accrualPeriodicities vector of class [character](#)

Method getDCAccrualPeriodicities(): Get DC accrual periodicities

Usage:

DCEntry\$getDCAccrualPeriodicities()

Returns: a list of objects of class [DCAccrualPeriodicity](#)

Method addDCAccrualPolicy(): Adds DC accrual policy

Usage:

DCEntry\$addDCAccrualPolicy(accrualPolicy)

Arguments:

accrualPolicy object of class [DCAccrualPolicy](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAccrualPolicy(): Deletes DC accrual policy

Usage:

DCEntry\$delDCAccrualPolicy(accrualPolicy)

Arguments:

accrualPolicy object of class [DCAccrualPolicy](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAccrualPolicies(): Set DC accrual policies

Usage:

DCEntry\$setDCAccrualPolicies(accrualPolicies)

Arguments:

accrualPolicies vector of class [character](#)

Method getDCAccrualPolicies(): Get DC accrual policies

Usage:

DCEntry\$getDCAccrualPolicies()

Returns: a list of objects of class [DCAccrualPolicy](#)

Method addDCAlternative(): Adds DC alternative

Usage:

DCEntry\$addDCAlternative(alternative)

Arguments:

alternative object of class [DCAlternative](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAlternative(): Deletes DC alternative

Usage:

DCEntry\$delDCAlternative(alternative)

Arguments:

alternative object of class [DCAlternative](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAlternatives(): Set DC alternatives

Usage:

DCEntry\$setDCAlternatives(alternatives)

Arguments:

alternatives vector of class [character](#)

Method getDCAlternatives(): Get DC alternatives

Usage:

DCEntry\$getDCAlternatives()

Returns: a list of objects of class [DCAlternative](#)

Method addDCAudience(): Adds DC audience

Usage:

DCEntry\$addDCAudience(audience)

Arguments:

audience object of class [DCAudience](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAudience(): Deletes DC audience

Usage:

DCEntry\$delDCAudience(audience)

Arguments:

audience object of class [DCAudience](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAudiences(): Set DC audiences

Usage:

DCEntry\$setDCAudiences(audiences)

Arguments:

audiences vector of class [character](#)

Method getDCAudiences(): Get DC audiences

Usage:

DCEntry\$getDCAudiences()

Returns: a list of objects of class [DCAudience](#)

Method addDCAvailable(): Adds DC available

Usage:

DCEntry\$addDCAvailable(available)

Arguments:

available object of class [DCAvailable](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCAvailable(): Deletes DC available

Usage:

DCEntry\$delDCAvailable(available)

Arguments:

available object of class [DCAvailable](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCAvailables(): Set DC availables

Usage:

DCEntry\$setDCAvailables(availables)

Arguments:

availables vector of class [character](#)

Method getDCAvailables(): Get DC availables

Usage:

DCEntry\$getDCAvailables()

Returns: a list of objects of class [DCAvailable](#)

Method addDCBibliographicCitation(): Adds DC bibliographic citation

Usage:

DCEntry\$addDCBibliographicCitation(bibliographicCitation)

Arguments:

bibliographicCitation object of class [DCBibliographicCitation](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCBibliographicCitation(): Deletes DC bibliographic citation

Usage:

DCEntry\$delDCBibliographicCitation(bibliographicCitation)

Arguments:

bibliographicCitation object of class [DCBibliographicCitation](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCBibliographicCitations(): Set bibliographic citations

Usage:

```
DCEntry$setDCBibliographicCitations(bibliographicCitations)
```

Arguments:

bibliographicCitations vector of class [character](#)

Method getDCBibliographicCitations(): Get bibliographic citations

Usage:

```
DCEntry$getDCBibliographicCitations()
```

Returns: the list of objects of class [DCBibliographicCitation](#)

Method addDCConformsTo(): Adds DC conforms to

Usage:

```
DCEntry$addDCConformsTo(conformsTo)
```

Arguments:

conformsTo object of class [DCConformsTo](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCConformsTo(): Deletes DC conforms to

Usage:

```
DCEntry$delDCConformsTo(conformsTo)
```

Arguments:

conformsTo object of class [DCConformsTo](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCConformsTo(): Set DC conforms to

Usage:

```
DCEntry$setDCConformsTo(conformsTo)
```

Arguments:

conformsTo vector of class [character](#)

Method getDCConformsTo(): Get DC conforms to

Usage:

```
DCEntry$getDCConformsTo()
```

Returns: the list of objects of class [DCConformsTo](#)

Method addDCContributor(): Adds DC contributor

Usage:

```
DCEntry$addDCContributor(contributor)
```

Arguments:

contributor object of class [DCContributor](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCContributor(): Deletes DC contributor

Usage:

DCEntry\$delDCContributor(contributor)

Arguments:

contributor object of class [DCContributor](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCContributors(): Set DC contributors

Usage:

DCEntry\$setDCContributors(contributors)

Arguments:

contributors vector of class [character](#)

Method getDCContributors(): Get DC contributors

Usage:

DCEntry\$getDCContributors()

Returns: list of objects of class [DCContributor](#)

Method addDCCoverage(): Adds DC coverage

Usage:

DCEntry\$addDCCoverage(coverage)

Arguments:

coverage object of class [DCCoverage](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCCoverage(): Deletes DC coverage

Usage:

DCEntry\$delDCCoverage(coverage)

Arguments:

coverage object of class [DCCoverage](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCCoverages(): Set DC coverages

Usage:

DCEntry\$setDCCoverages(coverages)

Arguments:

coverages coverages vector of class [character](#)

Method getDCCoverages(): Get DC coverages

Usage:

DCEntry\$getDCCoverages()

Returns: a list of objects of class [DCCoverage](#)

Method addDCCreated(): Adds DC created

Usage:

DCEntry\$addDCCreated(created)

Arguments:

created object of class [DCCreated](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCCreated(): Deletes DC created

Usage:

DCEntry\$delDCCreated(created)

Arguments:

created object of class [DCCreated](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method addDCCreator(): Adds DC creator

Usage:

DCEntry\$addDCCreator(creator)

Arguments:

creator object of class [DCCreator](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCCreator(): Deletes DC creator

Usage:

DCEntry\$delDCCreator(creator)

Arguments:

creator object of class [DCCreator](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCCreators(): Set DC creators

Usage:

DCEntry\$setDCCreators(creators)

Arguments:

creators creators

Method getDCCreators(): Get DC creators

Usage:

DCEntry\$getDCCreators()

Returns: a list of objects of class [DCCreator](#)

Method addDCDate(): Adds DC date

Usage:

DCEntry\$addDCDate(date)

Arguments:

date object of class [DCDate](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCDate(): Deletes DC date*Usage:*

```
DCEntry$delDCDate(date)
```

Arguments:

date object of class [DCDate](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCDates(): Set DC Creators*Usage:*

```
DCEntry$setDCDates(dates)
```

Arguments:

dates dates vector of class [Date](#) or [POSIXt](#)

Method getDCDates(): Get DC Dates*Usage:*

```
DCEntry$getDCDates()
```

Returns: a list of objects of class [DCDate](#)

Method addDCDateAccepted(): Adds DC date accepted*Usage:*

```
DCEntry$addDCDateAccepted(dateAccepted)
```

Arguments:

dateAccepted object of class [DCDateAccepted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCDateAccepted(): Deletes DC date accepted*Usage:*

```
DCEntry$delDCDateAccepted(dateAccepted)
```

Arguments:

dateAccepted object of class [DCDateAccepted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method addDCDateCopyrighted(): Adds DC date copyrighted*Usage:*

```
DCEntry$addDCDateCopyrighted(dateCopyrighted)
```

Arguments:

dateCopyrighted object of class [DCDateCopyrighted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCDateCopyrighted(): Deletes DC date copyrighted

Usage:

```
DCEntry$delDCDateCopyrighted(dateCopyrighted)
```

Arguments:

dateCopyrighted object of class [DCDateCopyrighted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method addDCDateSubmitted(): Adds DC date submitted

Usage:

```
DCEntry$addDCDateSubmitted(dateSubmitted)
```

Arguments:

dateSubmitted object of class [DCDateSubmitted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCDateSubmitted(): Deletes DC date submitted

Usage:

```
DCEntry$delDCDateSubmitted(dateSubmitted)
```

Arguments:

dateSubmitted object of class [DCDateSubmitted](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method addDCDescription(): Adds DC description

Usage:

```
DCEntry$addDCDescription(description)
```

Arguments:

description object of class [DCDescription](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCDescription(): Deletes DC description

Usage:

```
DCEntry$delDCDescription(description)
```

Arguments:

description object of class [DCDescription](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCDescriptions(): Set DC descriptions

Usage:

```
DCEntry$setDCDescriptions(descriptions)
```

Arguments:

descriptions vector of class [character](#)

Method getDCDescriptions(): Get DC descriptions

Usage:

```
DCEntry$getDCDescriptions()
```

Returns: a list of objects of class [DCDescription](#)

Method addDCEducationalLevel(): Adds DC educational level

Usage:

```
DCEntry$addDCEducationalLevel(educationalLevel)
```

Arguments:

educationalLevel object of class [DCEducationalLevel](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCEducationalLevel(): Deletes DC educational level

Usage:

```
DCEntry$delDCEducationalLevel(educationalLevel)
```

Arguments:

educationalLevel object of class [DCEducationalLevel](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCEducationalLevels(): set DC education levels

Usage:

```
DCEntry$setDCEducationalLevels(educationLevels)
```

Arguments:

educationLevels vector of class [character](#)

Method getDCEducationalLevels(): Get DC educational levels

Usage:

```
DCEntry$getDCEducationalLevels()
```

Returns: a list of objects of class [DCEducationalLevel](#)

Method addDCExtent(): Adds DC extent

Usage:

```
DCEntry$addDCExtent(extent)
```

Arguments:

extent object of class [DCExtent](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCExtent(): Deletes DC extent

Usage:

DCEntry\$delDCExtent(extent)

Arguments:

extent object of class [DCExtent](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCExtents(): Set DC extents

Usage:

DCEntry\$setDCExtents(extents)

Arguments:

extents vector of class [character](#)

Method getDCExtents(): Get DC extents

Usage:

DCEntry\$getDCExtents()

Returns: a list of objects of class [DCExtent](#)

Method addDCFormat(): Adds DC format

Usage:

DCEntry\$addDCFormat(format)

Arguments:

format object of class [DCFormat](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCFormat(): Deletes DC format

Usage:

DCEntry\$delDCFormat(format)

Arguments:

format object of class [DCFormat](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCFormats(): Set DC formats

Usage:

DCEntry\$setDCFormats(formats)

Arguments:

formats vector of class [character](#)

Method getDCFormats(): Get DC formats

Usage:

DCEntery\$getDCFormats()

Returns: a list of objects of class [DCFormat](#)

Method addDCHasPart(): Adds DC hasPart

Usage:

DCEntery\$addDCHasPart(hasPart)

Arguments:

hasPart object of class [DCHasPart](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCHasPart(): Deletes DC hasPart

Usage:

DCEntery\$delDCHasPart(hasPart)

Arguments:

hasPart object of class [DCHasPart](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCHasParts(): Set DC hasParts

Usage:

DCEntery\$setDCHasParts(hasParts)

Arguments:

hasParts vector of class [character](#)

Method getDCHasParts(): Get DC has part

Usage:

DCEntery\$getDCHasParts()

Returns: a list of objects of class [DCHasPart](#)

Method addDCHasVersion(): Adds DC hasVersion

Usage:

DCEntery\$addDCHasVersion(hasVersion)

Arguments:

hasVersion object of class [DCHasVersion](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCHasVersion(): Deletes DC hasVersion

Usage:

DCEntery\$delDCHasVersion(hasVersion)

Arguments:

hasVersion object of class [DCHasVersion](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCHasVersions(): Set DC hasVersions

Usage:

DCEntry\$setDCHasVersions(hasVersions)

Arguments:

hasVersions vector of class [character](#)

Method getDCHasVersions(): Get DC has versions

Usage:

DCEntry\$getDCHasVersions()

Returns: a list of objects of class [DCHasVersion](#)

Method addDCIdentifier(): Adds DC identifier

Usage:

DCEntry\$addDCIdentifier(identifier)

Arguments:

identifier object of class [DCIdentifier](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIdentifier(): Deletes DC identifier

Usage:

DCEntry\$delDCIdentifier(identifier)

Arguments:

identifier object of class [DCIdentifier](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIdentifiers(): Set DC identifiers

Usage:

DCEntry\$setDCIdentifiers(identifiers)

Arguments:

identifiers vector of class [character](#)

Method getDCIdentifiers(): Get DC identifiers

Usage:

DCEntry\$getDCIdentifiers()

Returns: a list of objects of class [DCIdentifier](#)

Method addDCInstructionalMethod(): Adds DC instructionalMethod

Usage:

DCEntry\$addDCInstructionalMethod(instructionalMethod)

Arguments:

instructionalMethod object of class [DCInstructionalMethod](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCInstructionalMethod(): Deletes DC instructionalMethod

Usage:

DCEntry\$delDCInstructionalMethod(instructionalMethod)

Arguments:

instructionalMethod object of class [DCInstructionalMethod](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCInstructionalMethods(): Set DC Instructional methods

Usage:

DCEntry\$setDCInstructionalMethods(instructionalMethods)

Arguments:

instructionalMethods vector of class [character](#)

Method getDCInstructionalMethods(): Get DC instructional methods

Usage:

DCEntry\$getDCInstructionalMethods()

Returns: a list of objects of class [DCInstructionalMethod](#)

Method addDCIsPartOf(): Adds DC isPartOf

Usage:

DCEntry\$addDCIsPartOf(isPartOf)

Arguments:

isPartOf object of class [DCIsPartOf](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIsPartOf(): Deletes DC isPartOf

Usage:

DCEntry\$delDCIsPartOf(isPartOf)

Arguments:

isPartOf object of class [DCIsPartOf](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIsPartOf(): Set DC IsPartOf

Usage:

DCEntry\$setDCIsPartOf(isPartOf)

Arguments:

isPartOf vector of class [character](#)

Method getDCIsPartOfs(): Get DC Is Part of

Usage:

DCEntry\$getDCIsPartOfs()

Returns: a list of objects of class [DCIsPartOf](#)

Method addDCIsReferencedBy(): Adds DC isReferencedBy

Usage:

DCEntry\$addDCIsReferencedBy(isReferencedBy)

Arguments:

isReferencedBy object of class [DCIsReferencedBy](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIsReferencedBy(): Deletes DC isReferencedBy

Usage:

DCEntry\$delDCIsReferencedBy(isReferencedBy)

Arguments:

isReferencedBy object of class [DCIsReferencedBy](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIsReferencedBys(): Set DC isReferencedBys

Usage:

DCEntry\$setDCIsReferencedBys(isReferencedBys)

Arguments:

isReferencedBys vector of class [character](#)

Method getDCIsReferencedBys(): Get DC Is Referenced by

Usage:

DCEntry\$getDCIsReferencedBys()

Returns: a list of objects of class [DCIsReferencedBy](#)

Method addDCIsReplacedBy(): Adds DC isReplacedBy

Usage:

DCEntry\$addDCIsReplacedBy(isReplacedBy)

Arguments:

isReplacedBy object of class [DCIsReplacedBy](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIsReplacedBy(): Deletes DC isReferencedBy

Usage:

DCEntry\$delDCIsReplacedBy(isReplacedBy)

Arguments:

isReplacedBy object of class [DCIsReplacedBy](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIsReplacedBy(): Set DC isReplacedBy

Usage:

DCEnter\$setDCIsReplacedBy(isReplacedBy)

Arguments:

isReplacedBy vector of class [character](#)

Method getDCIsReplacedBy(): Get DC Is Replaced by

Usage:

DCEnter\$getDCIsReplacedBy()

Returns: a list of objects of class [DCIsReplacedBy](#)

Method addDCIsRequiredBy(): Adds DC isRequiredBy

Usage:

DCEnter\$addDCIsRequiredBy(isRequiredBy)

Arguments:

isRequiredBy object of class [DCIsRequiredBy](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIsRequiredBy(): Deletes DC isRequiredBy

Usage:

DCEnter\$delDCIsRequiredBy(isRequiredBy)

Arguments:

isRequiredBy object of class [DCIsRequiredBy](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIsRequiredBy(): Set DC isRequiredBy

Usage:

DCEnter\$setDCIsRequiredBy(isRequiredBy)

Arguments:

isRequiredBy vector of class [character](#)

Method getDCIsRequiredBy(): Get DC Is Required by

Usage:

DCEnter\$getDCIsRequiredBy()

Returns: a list of objects of class [DCIsRequiredBy](#)

Method addDCIsVersionOf(): Adds DC isVersionOf

Usage:

DCEnter\$addDCIsVersionOf(isVersionOf)

Arguments:

isVersionOf object of class [DCIsVersionOf](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIsVersionOf(): Deletes DC isVersionOf

Usage:

DCEntry\$delDCIsVersionOf(isVersionOf)

Arguments:

isVersionOf object of class [DCIsVersionOf](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCIsVersionOfs(): Set DC isVersionOfs

Usage:

DCEntry\$setDCIsVersionOfs(isVersionOfs)

Arguments:

isVersionOfs vector of class [character](#)

Method getDCIsVersionOfs(): Get DC Is Version Of

Usage:

DCEntry\$getDCIsVersionOfs()

Returns: a list of objects of class [DCIsVersionOf](#)

Method addDCIssued(): Adds DC issued

Usage:

DCEntry\$addDCIssued(issued)

Arguments:

issued object of class [DCIssued](#) or vector of class [Date](#), [POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCIssued(): Deletes DC issued

Usage:

DCEntry\$delDCIssued(issued)

Arguments:

issued object of class [DCIssued](#) or vector of class [Date](#), [POSIXt](#) or [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method addDCLanguage(): Adds DC language

Usage:

DCEntry\$addDCLanguage(language)

Arguments:

language object of class [DCLanguage](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCLanguage(): Deletes DC language

Usage:

DCEntry\$delDCLanguage(language)

Arguments:

language object of class [DCLanguage](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCLanguages(): Set DC languages

Usage:

DCEntry\$setDCLanguages(languages)

Arguments:

languages languages vector of class [character](#)

Method getDCLanguages(): Get languages

Usage:

DCEntry\$getDCLanguages()

Returns: a list of objects of class [DCLanguage](#)

Method addDCLicense(): Adds DC license

Usage:

DCEntry\$addDCLicense(license)

Arguments:

license object of class [DCLicense](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCLicense(): Deletes DC license

Usage:

DCEntry\$delDCLicense(license)

Arguments:

license object of class [DCLicense](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCLicenses(): Set DC licences

Usage:

DCEntry\$setDCLicenses(licenses)

Arguments:

licenses vector of class [character](#)

Method getDCLicenses(): Get DC licenses

Usage:

DCEntry\$getDCLicenses()

Returns: a list of objects of class [DCLicense](#)

Method addDCMediator(): Adds DC mediator

Usage:

DCEntry\$addDCMediator(mediator)

Arguments:

mediator object of class [DCMediator](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCMediator(): Deletes DC mediator

Usage:

DCEntry\$delDCMediator(mediator)

Arguments:

mediator object of class [DCMediator](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCMediators(): Set DC mediators

Usage:

DCEntry\$setDCMediators(mediators)

Arguments:

mediators vector of class [character](#)

Method getDCMediators(): Get DC mediators

Usage:

DCEntry\$getDCMediators()

Returns: a list of objects of class [DCMediator](#)

Method addDCMedium(): Adds DC medium

Usage:

DCEntry\$addDCMedium(medium)

Arguments:

medium object of class [DCMedium](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCMedium(): Deletes DC medium

Usage:

DCEntry\$delDCMedium(medium)

Arguments:

medium object of class [DCMedium](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCMediums(): Set DC mediums

Usage:

DCEntry\$setDCMediums(mediums)

Arguments:

mediums vector of class [character](#)

Method getDCMediums(): Get DC mediums

Usage:

DCEntry\$getDCMediums()

Returns: a list of objects of class [DCMedium](#)

Method addDCModified(): Adds DC modified

Usage:

DCEntry\$addDCModified(modified)

Arguments:

modified object of class [DCModified](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCModified(): Deletes DC modified

Usage:

DCEntry\$delDCModified(modified)

Arguments:

modified object of class [DCModified](#) or vector of class [Date](#),[POSIXt](#) or [character](#) and length 1

Returns: TRUE if deletes, FALSE otherwise

Method addDCProvenance(): Adds DC provenance

Usage:

DCEntry\$addDCProvenance(provenance)

Arguments:

provenance object of class [DCProvenance](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCProvenance(): Deletes DC provenance

Usage:

DCEntry\$delDCProvenance(provenance)

Arguments:

provenance object of class [DCProvenance](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCProvenances(): Set DC provenances

Usage:

DCEntry\$setDCProvenances(provenances)

Arguments:

provenances vector of class [character](#)

Method getDCProvenances(): Get DC provenances

Usage:

DCEntry\$getDCProvenances()

Returns: a list of objects of class [DCProvenance](#)

Method addDCPublisher(): Adds DC publisher

Usage:

DCEntry\$addDCPublisher(publisher)

Arguments:

publisher object of class [DCPublisher](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCPublisher(): Deletes DC publisher

Usage:

DCEntry\$delDCPublisher(publisher)

Arguments:

publisher object of class [DCPublisher](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCPublishers(): Set DC publishers

Usage:

DCEntry\$setDCPublishers(publishers)

Arguments:

publishers vector of class [character](#)

Method getDCPublishers(): Get DC publishers

Usage:

DCEntry\$getDCPublishers()

Returns: a list of objects of class [DCPublisher](#)

Method addDCReferences(): Adds DC references

Usage:

DCEntry\$addDCReferences(references)

Arguments:

references object of class [DCReferences](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCReferences(): Deletes DC references

Usage:

DCEntry\$delDCReferences(references)

Arguments:

references object of class [DCReferences](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCReferences(): Set DC references

Usage:

```
DCEntry$setDCReferences(references)
```

Arguments:

references vector of class [character](#)

Method getDCReferences(): Get DC references

Usage:

```
DCEntry$getDCReferences()
```

Returns: a list of objects of class [DCReferences](#)

Method addDCRelation(): Adds DC relation

Usage:

```
DCEntry$addDCRelation(relation)
```

Arguments:

relation object of class [DCRelation](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCRelation(): Deletes DC relation

Usage:

```
DCEntry$delDCRelation(relation)
```

Arguments:

relation object of class [DCRelation](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCRelations(): Set DC relations

Usage:

```
DCEntry$setDCRelations(relations)
```

Arguments:

relations vector of class [character](#)

Method getDCRelations(): Get DC relations

Usage:

```
DCEntry$getDCRelations()
```

Returns: a list of objects of class [DCRelation](#)

Method addDCReplaces(): Adds DC replaces

Usage:

```
DCEntry$addDCReplaces(replaces)
```

Arguments:

replaces object of class [DCReplaces](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCReplaces(): Deletes DC replaces

Usage:

DCEntry\$delDCReplaces(replaces)

Arguments:

replaces object of class [DCReplaces](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCReplaces(): Set DC replaces

Usage:

DCEntry\$setDCReplaces(replaces)

Arguments:

replaces vector of class [character](#)

Method getDCReplaces(): Get DC replaces

Usage:

DCEntry\$getDCReplaces()

Returns: a list of objects of class [DCReplaces](#)

Method addDCRequires(): Adds DC requires

Usage:

DCEntry\$addDCRequires(requires)

Arguments:

requires object of class [DCRequires](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCRequires(): Deletes DC requires

Usage:

DCEntry\$delDCRequires(requires)

Arguments:

requires object of class [DCRequires](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCRequires(): Set DC requires

Usage:

DCEntry\$setDCRequires(requires)

Arguments:

requires vector of class [character](#)

Method getDCRequires(): Get DC requires

Usage:

DCEntry\$getDCRequires()

Returns: a list of objects of class [DCRequires](#)

Method addDCRights(): Adds DC rights

Usage:

DCEntry\$addDCRights(rights)

Arguments:

rights object of class [DCRights](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCRights(): Deletes DC rights

Usage:

DCEntry\$delDCRights(rights)

Arguments:

rights object of class [DCRights](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCRights(): Set DC rights

Usage:

DCEntry\$setDCRights(rights)

Arguments:

rights vector of class [character](#)

Method getDCRights(): Get DC rights

Usage:

DCEntry\$getDCRights()

Returns: a list of objects of class [DCRights](#)

Method addDCRightsHolder(): Adds DC rightsHolder

Usage:

DCEntry\$addDCRightsHolder(rightsHolder)

Arguments:

rightsHolder object of class [DCRightsHolder](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCRightsHolder(): Deletes DC rightsHolder

Usage:

DCEntry\$delDCRightsHolder(rightsHolder)

Arguments:

rightsHolder object of class [DCRightsHolder](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCRightsHolders(): Set DC rights holders

Usage:

DCEntry\$setDCRightsHolders(rightsHolders)

Arguments:

rightsHolders vector of class [character](#)

Method getDCRightsHolders(): Get DC rights holders

Usage:

DCEntry\$getDCRightsHolders()

Returns: a list of objects of class [DCRightsHolder](#)

Method addDCSource(): Adds DC source

Usage:

DCEntry\$addDCSource(source)

Arguments:

source object of class [DCSource](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCSource(): Deletes DC source

Usage:

DCEntry\$delDCSource(source)

Arguments:

source object of class [DCSource](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCSources(): Set DC sources

Usage:

DCEntry\$setDCSources(sources)

Arguments:

sources vector of class [character](#)

Method getDCSources(): Get DC sources

Usage:

DCEntry\$getDCSources()

Returns: a list of objects of class [DCSource](#)

Method addDCSubject(): Adds DC subject

Usage:

DCEntry\$addDCSubject(subject)

Arguments:

subject object of class [DCSubject](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method `delDCSubject()`: Deletes DC subject

Usage:

`DCEntry$delDCSubject(subject)`

Arguments:

subject object of class [DCSubject](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method `setDCSubjects()`: Set DC subjects

Usage:

`DCEntry$setDCSubjects(subjects)`

Arguments:

subjects vector of class [character](#)

Method `getDCSubjects()`: Get DC Subjects

Usage:

`DCEntry$getDCSubjects()`

Returns: a list of objects of class [DCSubject](#)

Method `addDCTableOfContents()`: Adds DC tableOfContents

Usage:

`DCEntry$addDCTableOfContents(tableOfContents)`

Arguments:

tableOfContents object of class [DCTableOfContents](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method `delDCTableOfContents()`: Deletes DC tableOfContents

Usage:

`DCEntry$delDCTableOfContents(tableOfContents)`

Arguments:

tableOfContents object of class [DCTableOfContents](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method `setDCTablesOfContents()`: Set DC tables of contents

Usage:

`DCEntry$setDCTablesOfContents(tablesOfContents)`

Arguments:

tablesOfContents vector of class [character](#)

Method getDCTablesOfContent(): Get DC tables of contents

Usage:

DCEntry\$getDCTablesOfContent()

Returns: a list of objects of class [DCTableOfContents](#)

Method addDCTemporal(): Adds DC temporal

Usage:

DCEntry\$addDCTemporal(temporal)

Arguments:

temporal object of class [DCTemporal](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCTemporal(): Deletes DC temporal

Usage:

DCEntry\$delDCTemporal(temporal)

Arguments:

temporal object of class [DCTemporal](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCTemporals(): Set DC temporals

Usage:

DCEntry\$setDCTemporals(temporals)

Arguments:

temporals vector of class [character](#)

Method getDCTemporals(): Get DC temporals

Usage:

DCEntry\$getDCTemporals()

Returns: a list of objects of class [DCTemporal](#)

Method addDCTitle(): Adds DC title

Usage:

DCEntry\$addDCTitle(title)

Arguments:

title object of class [DCTitle](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCTitle(): Deletes DC title

Usage:

DCEntry\$delDCTitle(title)

Arguments:

title object of class [DCTitle](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCTitles(): Set DC titles

Usage:

DCEntry\$setDCTitles(titles)

Arguments:

titles vector of class [character](#)

Method getDCTitles(): Get DC titles

Usage:

DCEntry\$getDCTitles()

Returns: a list of objects of class [DCTitle](#)

Method addDCType(): Adds DC type

Usage:

DCEntry\$addDCType(type)

Arguments:

type object of class [DCType](#) or vector of class [character](#) and length 1

Returns: TRUE if added, FALSE otherwise

Method delDCType(): Deletes DC type

Usage:

DCEntry\$delDCType(type)

Arguments:

type object of class [DCType](#) or vector of class [character](#) and length 1

Returns: TRUE if deleted, FALSE otherwise

Method setDCTypes(): Set DC Types

Usage:

DCEntry\$setDCTypes(types)

Arguments:

types vector of class [character](#)

Method getDCTypes(): Get DC types

Usage:

DCEntry\$getDCTypes()

Returns: a list of objects of class [DCType](#)

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

Usage:

DCEntry\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Examples

```
#encoding
dcentry <- DCEntry$new()
dcentry$setId("my-dc-entry")

#fill dc entry
dcentry$addDCDate(Sys.time())
dcentry$addDCTitle("atom4R - Tools to read/write and publish metadata as Atom XML format")
dcentry$addDCType("Software")
creator <- DCCreator$new(value = "Blondel, Emmanuel")
creator$attrs[["affiliation"]] <- "Independent"
dcentry$addDCCreator(creator)
dcentry$addDCSubject("R")
dcentry$addDCSubject("FAIR")
dcentry$addDCSubject("Interoperability")
dcentry$addDCSubject("Open Science")
dcentry$addDCDescription("Atom4R offers tools to read/write and publish metadata as Atom XML")
dcentry$addDCPublisher("GitHub")
funder <- DCContributor$new(value = "CNRS")
funder$attrs[["type"]] <- "Funder"
dcentry$addDCContributor(funder)
dcentry$addDCRelation("Github repository: https://github.com/eblondel/atom4R")
dcentry$addDCSource("Atom Syndication format - https://www.ietf.org/rfc/rfc4287")
dcentry$addDCSource("AtomPub, The Atom publishing protocol - https://tools.ietf.org/html/rfc5023")
dcentry$addDCSource("Sword API - http://swordapp.org/")
dcentry$addDCSource("Dublin Core Metadata Initiative - https://www.dublincore.org/")
dcentry$addDCSource("Guidelines for implementing Dublin Core in XML")
dcentry$addDCLicense("NONE")
dcentry$addDCRights("MIT License")
dcentry$addDCHasPart("part1")
dcentry$addDCHasPart("part2")
dcentry$addDCHasVersion("0.2")
dcentry$addDCIsPartOf("CRAN")
dcentry$addDCIsPartOf("GitHub")
dcentry$addDCIsReferencedBy("CRAN")
dcentry$addDCIsReferencedBy("GitHub")
dcentry$addDCIsRequiredBy("zen4R")
dcentry$addDCIsRequiredBy("cloud4R")

xml <- dcentry$encode()

#decoding
dcentry2 <- DCEntry$new(xml = xml)
xml2 <- dcentry2$encode()
```

DCExtent

DCExtent

Description

This class models an DublinCore 'extent' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'extent' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCFormat](#)
-> DCExtent

Methods

Public methods:

- [DCExtent\\$new\(\)](#)
- [DCExtent\\$clone\(\)](#)

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

Usage:

`DCExtent$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**
`value` value

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

Usage:

`DCExtent$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/extent>

DCFormat

DCFormat

Description

This class models an DublinCore 'format' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'format' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCFormat

Methods

Public methods:

- [DCFormat\\$new\(\)](#)
- [DCFormat\\$clone\(\)](#)

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

Usage:

```
DCFormat$new(xml = NULL, term = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/format>

DCHasPart

DCHasPart

Description

This class models an DublinCore 'hasPart' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'hasPart' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCHasPart

Methods

Public methods:

- [DCHasPart\\$new\(\)](#)
- [DCHasPart\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'hasPart' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCHasPart$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCHasVersion

DCHasVersion

Description

This class models an DublinCore 'hasVersion' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'hasPart' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCHasVersion

Methods

Public methods:

- [DCHasVersion\\$new\(\)](#)
- [DCHasVersion\\$clone\(\)](#)

Method `new()`: This method is used to create an Dublin core 'hasVersion' element. Use `dc` to `TRUE` to use Dublin core namespace instead of DC terms.

Usage:

```
DCHasVersion$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

`dc` use DC namespace?

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCIdentifier

DCIdentifier

Description

This class models an DublinCore 'identifier' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'identifier' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIdentifier

Methods**Public methods:**

- [DCIdentifier\\$new\(\)](#)
- [DCIdentifier\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'identifier' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIdentifier$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/identifier>

DCInstructionalMethod *DCInstructionalMethod*

Description

This class models an DublinCore 'instructionalMethod' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core 'instructionalMethod' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCInstructionalMethod

Methods

Public methods:

- [DCInstructionalMethod\\$new\(\)](#)
- [DCInstructionalMethod\\$clone\(\)](#)

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

Usage:

```
DCInstructionalMethod$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/instructionalMethod>

DCIsPartOf

DCIsPartOf

Description

This class models an DublinCore 'isPartOf' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'isPartOf' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIsPartOf

Methods

Public methods:

- [DCIsPartOf\\$new\(\)](#)
- [DCIsPartOf\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'isPartOf' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIsPartOf$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCIsReferencedBy	<i>DCIsReferencedBy</i>
------------------	-------------------------

Description

This class models an DublinCore 'isReferencedBy' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'isReferencedBy' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIsReferencedBy

Methods

Public methods:

- [DCIsReferencedBy\\$new\(\)](#)
- [DCIsReferencedBy\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'isReferencedBy' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIsReferencedBy$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCIsReplacedBy	<i>DCIsReplacedBy</i>
----------------	-----------------------

Description

This class models an DublinCore 'isReplacedBy' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'isReplacedBy' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIsReplacedBy

Methods

Public methods:

- [DCIsReplacedBy\\$new\(\)](#)
- [DCIsReplacedBy\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'isReplacedBy' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIsReplacedBy$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
 value value
 dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCIsRequiredBy	<i>DCIsRequiredBy</i>
----------------	-----------------------

Description

This class models an DublinCore 'isRequiredBy' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'isRequiredBy' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIsRequiredBy

Methods**Public methods:**

- [DCIsRequiredBy\\$new\(\)](#)
- [DCIsRequiredBy\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'isRequiredBy' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIsRequiredBy$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value
dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCIssued

DCIssued

Description

This class models an DublinCore 'issued' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'issued' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCIssued](#)

Methods

Public methods:

- [DCIssued\\$new\(\)](#)
- [DCIssued\\$clone\(\)](#)

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

Usage:

[DCIssued\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

[DCIssued\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/issued>

DCIsVersionOf	<i>DCIsVersionOf</i>
---------------	----------------------

Description

This class models an DublinCore 'isVersionOf' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'isVersionOf' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCIsVersionOf

Methods

Public methods:

- [DCIsVersionOf\\$new\(\)](#)
- [DCIsVersionOf\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'isVersionOf' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCIsVersionOf$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value
dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/#http://purl.org/dc/terms>

DCLanguage

DCLanguage

Description

This class models an DublinCore 'language' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'language' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCLanguage

Methods**Public methods:**

- [DCLanguage\\$new\(\)](#)
- [DCLanguage\\$clone\(\)](#)

Method `new()`: This method is used to create an Dublin core 'language' element. Use `dc` to `TRUE` to use Dublin core namespace instead of DC terms.

Usage:

```
DCLanguage$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

`xml` object of class [XMLInternalNode-class](#) from [XML](#)
`value` value
`dc` use DC namespace?

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/language>

DCLicense

DCLicense

Description

This class models an DublinCore 'license' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'license' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRights](#)
-> [DCLicense](#)

Methods

Public methods:

- [DCLicense\\$new\(\)](#)
- [DCLicense\\$clone\(\)](#)

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

Usage:

[DCLicense\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

[DCLicense\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/license>

DCMediator

DCMediator

Description

This class models an DublinCore 'mediator' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'mediator' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCAudience](#)
-> DCMediator

Methods

Public methods:

- [DCMediator\\$new\(\)](#)
- [DCMediator\\$clone\(\)](#)

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

Usage:

```
DCMediator$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/mediator>

DCMedium

DCMedium

Description

This class models an DublinCore 'medium' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'medium' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCFormat](#)
-> DCMedium

Methods

Public methods:

- [DCMedium\\$new\(\)](#)
- [DCMedium\\$clone\(\)](#)

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

Usage:

`DCMedium$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from [XML](#)
`value` value

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

Usage:

`DCMedium$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/medium>

DCMIVocabulary

DCMI Vocabulary class

Description

This class models an DCMI Vocabulary

Format

[R6Class](#) object.

Details

DCMIVocabulary

Value

Object of [R6Class](#) for modelling an Dublin Core element

Public fields

id id

doc doc

representation representation

data data

Methods

Public methods:

- [DCMIVocabulary\\$new\(\)](#)
- [DCMIVocabulary\\$fetch\(\)](#)
- [DCMIVocabulary\\$clone\(\)](#)

Method `new()`: This method is used to read a DCMI vocabulary RDF doc. The format corresponds to the RDF format as used by **rdflib** `rdf_parse` function.

Usage:

```
DCMIVocabulary$new(id, doc, format, fetch = TRUE)
```

Arguments:

id id

doc doc

format format

fetch fetch

Method `fetch()`: Runs a Sparql query over the RDF vocabulary to fetch the vocabulary content.

Usage:

DCMIVocabulary\$fetch()

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

Usage:

DCMIVocabulary\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

DCModified

DCModified

Description

This class models an DublinCore 'modified' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'modified' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCModified](#)

Methods

Public methods:

- [DCModified\\$new\(\)](#)
- [DCModified\\$clone\(\)](#)

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

Usage:

DCModified\$new(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/modified>

DCProvenance

DCProvenance

Description

This class models an DublinCore 'provenance' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'provenance' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCProvenance

Methods**Public methods:**

- [DCProvenance\\$new\(\)](#)
- [DCProvenance\\$clone\(\)](#)

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

Usage:

```
DCProvenance$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/provenance>

DCPublisher

DCPublisher

Description

This class models an DublinCore 'publisher' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'publisher' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCPublisher

Methods

Public methods:

- [DCPublisher\\$new\(\)](#)
- [DCPublisher\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'publisher' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCPublisher$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/publisher>

DCReferences

DCReferences

Description

This class models an DublinCore 'references' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'references' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRelation](#)
-> DCReferences

Methods

Public methods:

- [DCReferences\\$new\(\)](#)
- [DCReferences\\$clone\(\)](#)

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

Usage:

`DCReferences$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

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

Usage:

`DCReferences$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/references>

DCRelation	<i>DCRelation</i>
------------	-------------------

Description

This class models an DublinCore 'relation' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'relation' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCRelation

Methods

Public methods:

- [DCRelation\\$new\(\)](#)
- [DCRelation\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'relation' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCRelation$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/relation>

DCReplaces

DCReplaces

Description

This class models an DublinCore 'replaces' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'replaces' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRelation](#)
-> DCReplaces

Methods**Public methods:**

- [DCReplaces\\$new\(\)](#)
- [DCReplaces\\$clone\(\)](#)

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

Usage:

```
DCReplaces$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/replaces>

DCRequires

DCRequires

Description

This class models an DublinCore 'requires' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'requires' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRelation](#)
-> DCRequires

Methods

Public methods:

- [DCRequires\\$new\(\)](#)
- [DCRequires\\$clone\(\)](#)

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

Usage:

`DCRequires$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

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

Usage:

`DCRequires$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/requires>

DCRights

*DCRights***Description**

This class models an DublinCore 'rights' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'rights' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCRights

Methods**Public methods:**

- [DCRights\\$new\(\)](#)
- [DCRights\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'rights' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCRights$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**
 term term
 value value
 dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/rights>

DCRightsHolder	<i>DCRightsHolder</i>
----------------	-----------------------

Description

This class models an DublinCore 'rightsHolder' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'rightsHolder' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCRightsHolder

Methods**Public methods:**

- [DCRightsHolder\\$new\(\)](#)
- [DCRightsHolder\\$clone\(\)](#)

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

Usage:

```
DCRightsHolder$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/rightsHolder>

DCSource

DCSource

Description

This class models an DublinCore 'source' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'source' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCRelation](#)
-> [DCSource](#)

Methods

Public methods:

- [DCSource\\$new\(\)](#)
- [DCSource\\$clone\(\)](#)

Method [new\(\)](#): This method is used to create an Dublin core 'source' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCSource$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/source>

DCSpatial

DCSpatial

Description

This class models an DublinCore 'spatial' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'spatial' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCCoverage](#)
-> [DCSpatial](#)

Methods

Public methods:

- [DCSpatial\\$new\(\)](#)
- [DCSpatial\\$clone\(\)](#)

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

Usage:

[DCSpatial\\$new](#)(xml = NULL, value = NULL)

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)

value value

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

Usage:

[DCSpatial\\$clone](#)(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/spatial>

DCSubject

DCSubject

Description

This class models an DublinCore 'subject' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'subject' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCSubject

Methods

Public methods:

- [DCSubject\\$new\(\)](#)
- [DCSubject\\$clone\(\)](#)

Method `new()`: This method is used to create an Dublin core 'subject' element. Use `dc` to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCSubject$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

`xml` object of class [XMLInternalNode-class](#) from [XML](#)

`value` value

`dc` use DC namespace?

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/subject>

DCTableOfContents	<i>DCTableOfContents</i>
-------------------	--------------------------

Description

This class models an DublinCore 'tableOfContents' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'tableOfContents' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDescription](#)
-> [DCTableOfContents](#)

Methods

Public methods:

- [DCTableOfContents\\$new\(\)](#)
- [DCTableOfContents\\$clone\(\)](#)

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

Usage:

```
DCTableOfContents$new(xml = NULL, value = NULL)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

value value

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/tableOfContents>

DCTemporal

DCTemporal

Description

This class models an DublinCore 'temporal' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'temporal' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCCoverage](#)
-> DCTemporal

Methods

Public methods:

- [DCTemporal\\$new\(\)](#)
- [DCTemporal\\$clone\(\)](#)

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

Usage:

`DCTemporal$new(xml = NULL, value = NULL)`

Arguments:

`xml` object of class [XMLInternalNode-class](#) from **XML**

`value` value

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

Usage:

`DCTemporal$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/temporal>

DCTitle

DCTitle

Description

This class models an DublinCore 'title' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'title' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCTitle

Methods

Public methods:

- [DCTitle\\$new\(\)](#)
- [DCTitle\\$clone\(\)](#)

Method new(): This method is used to create an Dublin core 'title' element. Use dc to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCTitle$new(xml = NULL, term = NULL, value = NULL, dc = FALSE)
```

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

term term

value value

dc use DC namespace?

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/title>

DCType

DCType

Description

This class models an DublinCore 'type' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'type' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> DCType

Methods

Public methods:

- [DCType\\$new\(\)](#)
- [DCType\\$clone\(\)](#)

Method `new()`: This method is used to create an Dublin core 'type' element. Use `dc` to TRUE to use Dublin core namespace instead of DC terms.

Usage:

```
DCType$new(xml = NULL, value = NULL, dc = FALSE)
```

Arguments:

`xml` object of class [XMLInternalNode-class](#) from [XML](#)

`value` value

`dc` use DC namespace?

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/type>

DCValid

DCValid

Description

This class models an DublinCore 'valid' element

Format

[R6Class](#) object.

Value

Object of [R6Class](#) for modelling an Dublin Core Terms 'valid' element

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomAbstractObject](#) -> [atom4R::DCElement](#) -> [atom4R::DCDate](#)
-> [DCValid](#)

Methods

Public methods:

- [DCValid\\$new\(\)](#)
- [DCValid\\$clone\(\)](#)

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

Usage:

`DCValid$new(xml = NULL, value = NULL)`

Arguments:

xml object of class [XMLInternalNode-class](#) from [XML](#)
value value

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

Usage:

`DCValid$clone(deep = FALSE)`

Arguments:

deep Whether to make a deep clone.

References

Dublin Core Metadata Initiative. <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/valid>

<code>getAtomClasses</code>	<i>getAtomClasses</i>
-----------------------------	-----------------------

Description

get the list of Atom classes, ie classes extending [AtomAbstractObject](#) super class, including classes eventually defined outside **atom4R**. In case the latter is on the search path, the list of Atom classes will be cached for optimized used by **atom4R** encoder/decoder.

Usage

```
getAtomClasses()
```

Author(s)

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

Examples

```
getAtomClasses()
```

<code>getAtomNamespace</code>	<i>getAtomNamespace</i>
-------------------------------	-------------------------

Description

`getAtomNamespace` gets a namespace given its id

Usage

```
getAtomNamespace(id)
```

Arguments

<code>id</code>	namespace prefix
-----------------	------------------

Author(s)

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

Examples

```
getAtomNamespace("GMD")
```

`getAtomNamespaces` *getAtomNamespaces*

Description

`getAtomNamespaces` gets the list of namespaces registered

Usage

`getAtomNamespaces()`

Author(s)

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

Examples

`getAtomNamespaces()`

`getAtomSchemas` *getAtomSchemas*

Description

`getAtomSchemas` gets the schemas registered in **atom4R**

Usage

`getAtomSchemas()`

Author(s)

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

Examples

`getAtomSchemas()`

getClassesInheriting *getClassesInheriting*

Description

get the list of classes inheriting a given super class provided by its name

Usage

```
getClassesInheriting(classname, extended, pretty)
```

Arguments

classname	the name of the superclass for which inheriting sub-classes have to be listed
extended	whether we want to look at user namespace for third-party sub-classes
pretty	prettify the output as data.frame

Examples

```
getClassesInheriting("DCElement")
```

getDCMIVocabularies *getDCMIVocabularies*

Description

getDCMIVocabularies allows to get the list of DCMI Vocabularies registered in **atom4R**

Usage

```
getDCMIVocabularies()
```

Author(s)

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

Examples

```
getDCMIVocabularies()
```

getDCMIVocabulary *getDCMIVocabulary*

Description

getDCMIVocabulary allows to get a registered DCMI Vocabulary by id registered in **atom4R**

Usage

```
getDCMIVocabulary(id)
```

Arguments

id identifier of the vocabulary

Author(s)

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

Examples

```
getDCMIVocabulary(id = "http://purl.org/dc/dcmitype/")
```

readDCEntry *readDCEntry*

Description

readDCEntry is a function to read a DC XML entry from a file or url into an object in the **atom4R** model.

Usage

```
readDCEntry(file, url, raw)
```

Arguments

file a valid file path, as object of class character
url a valid URL, as object of class character
raw indicates if the function should return the raw XML. By default this is set to FALSE and the function will try to map the xml data to the **atom4R** data model.

Value

a **atom4R** object inheriting DCEntry

Author(s)

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

Examples

```
dcfile <- paste0(
  "https://raw.githubusercontent.com/eblondel/atom4R/master/",
  "inst/extdata/examples/zenodo_dc_export.xml"
)
dc <- readDCEntry(dcfile)
```

registerAtomNamespace *registerAtomNamespace*

Description

registerAtomNamespace allows to register a new namespace in **atom4R**

Usage

```
registerAtomNamespace(id, uri, force)
```

Arguments

id	prefix of the namespace
uri	URI of the namespace
force	logical parameter indicating if registration has to be forced in case the identified namespace is already registered

Author(s)

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

Examples

```
registerAtomNamespace(id = "myprefix", uri = "http://someuri")
```

registerAtomSchema *registerAtomSchema*

Description

registerAtomSchema allows to register a new schema in **atom4R**

Usage

```
registerAtomSchema(xsdFile)
```

Arguments

xsdFile the schema XSD file

Author(s)

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

Examples

```
registerAtomSchema(xsdFile = "https://jvndb.jvn.jp/schema/atom.xsd")
```

setAtomNamespaces *setMetadataNamespaces*

Description

setMetadataNamespaces

Usage

```
setAtomNamespaces()
```

setAtomSchemas *setAtomSchemas*

Description

setAtomSchemas

Usage

```
setAtomSchemas()
```

setDCMIVocabularies *setDCMIVocabularies*

Description

setDCMIVocabularies

Usage

setDCMIVocabularies()

SwordClient

SwordClient class

Description

This class models an Sword service client

Format

[R6Class](#) object.

Details

SwordClient

Value

Object of [R6Class](#) for modelling an Sword client

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomPubClient](#) -> [SwordClient](#)

Methods**Public methods:**

- [SwordClient\\$new\(\)](#)
- [SwordClient\\$getServiceDocument\(\)](#)
- [SwordClient\\$getCollectionMembers\(\)](#)
- [SwordClient\\$clone\(\)](#)

Method new(): This method is to instantiate an Sword Client. By default the version is set to "2".

The `keyring_backend` can be set to use a different backend for storing the SWORD API user token with **keyring** (Default value is 'env').

The logger allows to specify the level of log (default is NULL), either "INFO" for **atom4R** logs or "DEBUG" for verbose HTTP client (curl) logs.

Usage:

```
SwordClient$new(
  url,
  version = "2",
  user = NULL,
  pwd = NULL,
  token = NULL,
  logger = NULL,
  keyring_backend = "env"
)
```

Arguments:

```
url url
version version. Default is "2"
user user
pwd pwd
token token
logger logger
keyring_backend keyring backend. Default is 'env'
```

Method getServiceDocument(): Get service document

Usage:

```
SwordClient$getServiceDocument(force = FALSE)
```

Arguments:

```
force force Force getting/refreshing of service document
```

Returns: object of class [SwordServiceDocument](#)

Method getCollectionMembers(): Get collection members. Unimplemented abstract method at [SwordClient](#) level

Usage:

```
SwordClient$getCollectionMembers()
```

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

Usage:

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

Arguments:

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

Note

Abstract class

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

SwordDataverseClient *SWORD Dataverse client class*

Description

This class models an Sword service Dataverse-specific API client

Format

[R6Class](#) object.

Details

SwordDataverseClient

Value

Object of [R6Class](#) for modelling an Sword Dataverse-specific APIclient

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomPubClient](#) -> [atom4R::SwordClient](#) -> SwordDataverseClient

Methods**Public methods:**

- [SwordDataverseClient\\$new\(\)](#)
- [SwordDataverseClient\\$getServiceDocument\(\)](#)
- [SwordDataverseClient\\$getCollectionMembers\(\)](#)
- [SwordDataverseClient\\$getDataverses\(\)](#)
- [SwordDataverseClient\\$getDataverse\(\)](#)
- [SwordDataverseClient\\$editDataverseEntry\(\)](#)
- [SwordDataverseClient\\$getDataverseRecord\(\)](#)
- [SwordDataverseClient\\$createDataverseRecord\(\)](#)
- [SwordDataverseClient\\$updateDataverseRecord\(\)](#)
- [SwordDataverseClient\\$deleteDataverseRecord\(\)](#)
- [SwordDataverseClient\\$publishDataverseRecord\(\)](#)
- [SwordDataverseClient\\$addFilesToDataverseRecord\(\)](#)

- [SwordDataverseClient\\$deleteFilesFromDataverseRecord\(\)](#)
- [SwordDataverseClient\\$clone\(\)](#)

Method new(): This method is to instantiate an Sword API Dataverse-specific Client.

The keyring_backend can be set to use a different backend for storing the SWORD DataVerse API user token with **keyring** (Default value is 'env').

The logger allows to specify the level of log (default is NULL), either "INFO" for **atom4R** logs or "DEBUG" for verbose HTTP client (curl) logs.

Usage:

```
SwordDataverseClient$new(
  hostname,
  token = NULL,
  logger = NULL,
  keyring_backend = "env"
)
```

Arguments:

hostname host name
 token token
 logger logger
 keyring_backend keyring backend. Default is 'env'

Method getServiceDocument(): Get service document

Usage:

```
SwordDataverseClient$getServiceDocument(force = FALSE)
```

Arguments:

force force Force getting/refreshing of service document

Returns: object of class [SwordServiceDocument](#)

Method getCollectionMembers(): Get collection members

Usage:

```
SwordDataverseClient$getCollectionMembers(collectionId)
```

Arguments:

collectionId collection ID

Returns: a list of [AtomFeed](#)

Method getDataverses(): Get dataverses. Equivalent to listCollections() from [Atom-PubClient](#)

Usage:

```
SwordDataverseClient$getDataverses(pretty = FALSE)
```

Arguments:

pretty prettify output as data.frame. Default is FALSE

Returns: an object of class data.frame

Method `getDataverse():` Get dataverse members by dataverse name. Equivalent to `getCollectionMembers()`

Usage:

`SwordDataverseClient$getDataverse(dataverse)`

Arguments:

`dataverse` dataverse name

Returns: a list of [AtomFeed](#)

Method `editDataverseEntry():` Edits a dataverse entry

Usage:

`SwordDataverseClient$editDataverseEntry(identifier)`

Arguments:

`identifier` identifier

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

Method `getDataverseRecord():` Get dataverse record

Usage:

`SwordDataverseClient$getDataverseRecord(identifier)`

Arguments:

`identifier` identifier

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

Method `createDataverseRecord():` Creates a dataverse record

Usage:

`SwordDataverseClient$createDataverseRecord(dataverse, entry)`

Arguments:

`dataverse` dataverse name

`entry` entry

the created [AtomEntry](#)

Method `updateDataverseRecord():` Updates a dataverse record

Usage:

`SwordDataverseClient$updateDataverseRecord(dataverse, entry, identifier)`

Arguments:

`dataverse` dataverse name

`entry` entry

`identifier` identifier of the entry to update

the created [AtomEntry](#)

Method `deleteDataverseRecord():` Deletes a dataverse record

Usage:

`SwordDataverseClient$deleteDataverseRecord(identifier)`

Arguments:

identifier identifier

Returns: TRUE if deleted, or returns an error otherwise

Method publishDataverseRecord(): Publishes a dataverse record

Usage:

SwordDataverseClient\$publishDataverseRecord(identifier)

Arguments:

identifier identifier

Returns: the published [AtomEntry](#)

Method addFilesToDataverseRecord(): Add files to a dataverse record

Usage:

SwordDataverseClient\$addFilesToDataverseRecord(identifier, files)

Arguments:

identifier identifier

files files

Method deleteFilesFromDataverseRecord(): Deletes files from a Dataverse record

Usage:

SwordDataverseClient\$deleteFilesFromDataverseRecord(identifier, files = NULL)

Arguments:

identifier identifier

files files

Returns: an object of class data.frame giving each file and it's deletion status

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

Usage:

SwordDataverseClient\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

Examples

```
## Not run:
#connect to SWORD Dataverse API
SWORD <- SwordDataverseClient$new(
  hostname = "localhost:8085",
  token = "<token>",
  logger = "DEBUG"
```

```

)

#for detailed operations check the wiki at:
#https://github.com/eblondel/atom4R/wiki#atom4R-publish-sword-dataverse

## End(Not run)

```

SwordHalClient	<i>SwordHalClient class</i>
----------------	-----------------------------

Description

This class models an Sword service client for HAL (Archives Houvertes)

Format

[R6Class](#) object.

Details

SwordHalClient

Value

Object of [R6Class](#) for modelling an Sword client

Super classes

[atom4R::atom4RLogger](#) -> [atom4R::AtomPubClient](#) -> [atom4R::SwordClient](#) -> SwordHalClient

Methods

Public methods:

- [SwordHalClient\\$new\(\)](#)
- [SwordHalClient\\$getServiceDocument\(\)](#)
- [SwordHalClient\\$getCollectionMembers\(\)](#)
- [SwordHalClient\\$clone\(\)](#)

Method [new\(\)](#): This method is to instantiate an Sword HAL (Archive Ouvertes - <https://hal.archives-ouvertes.fr/>) Client. By default the version is set to "2".

The `keyring_backend` can be set to use a different backend for storing the SWORD API user token with **keyring** (Default value is 'env').

The logger allows to specify the level of log (default is NULL), either "INFO" for **atom4R** logs or "DEBUG" for verbose HTTP client (curl) logs.

Usage:

```
SwordHalClient$new(  
  url,  
  user = NULL,  
  pwd = NULL,  
  logger = NULL,  
  keyring_backend = "env"  
)
```

Arguments:

url url

user user

pwd pwd

logger logger

keyring_backend keyring backend. Default value is 'env'

Method getServiceDocument(): Get service document

Usage:

```
SwordHalClient$getServiceDocument(force = FALSE)
```

Arguments:

force force Force getting/refreshing of service document

Returns: object of class [SwordServiceDocument](#)

Method getCollectionMembers(): Get collection members

Usage:

```
SwordHalClient$getCollectionMembers(collectionId)
```

Arguments:

collectionId collection ID

Returns: a list of [AtomFeed](#)

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

Note

Experimental

Author(s)

Emmanuel Blondel <emmanuel.blondell@gmail.com>

SwordServiceDocument *SwordServiceDocument class*

Description

This class models an Sword service document

Format

[R6Class](#) object.

Details

SwordServiceDocument

Value

Object of [R6Class](#) for modelling an Sword service document

Super class

[atom4R::atom4RLogger](#) -> SwordServiceDocument

Public fields

title title

collections collections

Methods

Public methods:

- [SwordServiceDocument\\$new\(\)](#)
- [SwordServiceDocument\\$getTitle\(\)](#)
- [SwordServiceDocument\\$getCollections\(\)](#)
- [SwordServiceDocument\\$clone\(\)](#)

Method new(): Initializes a [SwordServiceDocument](#) from XML

Usage:

`SwordServiceDocument$new(xml, logger = NULL)`

Arguments:

xml object of class [XMLInternalNode-class](#) from **XML**

logger logger

Method getTitle(): Get title

Usage:

`SwordServiceDocument$getTitle()`

Returns: object of class character

Method getCollections(): Get collections

Usage:

SwordServiceDocument\$getCollections()

Returns: object of class character

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

Usage:

SwordServiceDocument\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

Note

class used internally by **atom4R**

Author(s)

Emmanuel Blondel <emmanuel.blondel1@gmail.com>

Index

- * **'abstract'**
 - DCAbstract, [36](#)
- * **'accessRights'**
 - DCAccessRights, [37](#)
- * **'accrualMethod'**
 - DCAccrualMethod, [38](#)
- * **'accrualPeriodicity'**
 - DCAccrualPeriodicity, [39](#)
- * **'accrualPolicy'**
 - DCAccrualPolicy, [40](#)
- * **'alternative'**
 - DCAAlternative, [41](#)
- * **'audience'**
 - DCAudience, [42](#)
- * **'available'**
 - DCAvailable, [43](#)
- * **'bibliographicCitation'**
 - DCBibliographicCitation, [44](#)
- * **'conformsTo'**
 - DCConformsTo, [45](#)
- * **'contributor'**
 - DCCContributor, [46](#)
- * **'coverage'**
 - DCCoverage, [47](#)
- * **'creator'**
 - DCCreator, [49](#)
- * **'date'**
 - DCCreated, [48](#)
 - DCDate, [50](#)
- * **'dateAccepted'**
 - DCDateAccepted, [51](#)
- * **'dateCopyrighted'**
 - DCDateCopyrighted, [52](#)
- * **'dateSubmitted'**
 - DCDateSubmitted, [53](#)
- * **'description'**
 - DCDescription, [54](#)
- * **'educationalLevel'**
 - DCEducationalLevel, [55](#)
- * **'extent'**
 - DCExtent, [94](#)
- * **'format'**
 - DCFormat, [95](#)
- * **'hasPart'**
 - DCHasPart, [96](#)
- * **'hasVersion'**
 - DCHasVersion, [97](#)
- * **'identifier'**
 - DCIdentifier, [98](#)
- * **'instructionalMethod'**
 - DCInstructionalMethod, [99](#)
- * **'isPartOf'**
 - DCIsPartOf, [100](#)
- * **'isReferencedBy'**
 - DCIsReferencedBy, [101](#)
- * **'isReplacedBy'**
 - DCIsReplacedBy, [102](#)
- * **'isRequiredBy'**
 - DCIsRequiredBy, [103](#)
- * **'isVersionOf'**
 - DCIsVersionOf, [105](#)
- * **'issued'**
 - DCIssued, [104](#)
- * **'language'**
 - DCLanguage, [106](#)
- * **'license'**
 - DCLicense, [107](#)
- * **'mediator'**
 - DCMediator, [108](#)
- * **'medium'**
 - DCMedium, [109](#)
- * **'modified'**
 - DCModified, [111](#)
- * **'provenance'**
 - DCProvenance, [112](#)
- * **'publisher'**
 - DCPublisher, [113](#)
- * **'references'**

- DCReferences, [114](#)
- * **'relation'**
 - DCRelation, [115](#)
- * **'replaces'**
 - DCReplaces, [116](#)
- * **'requires'**
 - DCRequires, [117](#)
- * **'rights'**
 - DCRights, [118](#)
- * **'rightsHolder'**
 - DCRightsHolder, [119](#)
- * **'source'**
 - DCSource, [120](#)
- * **'spatial'**
 - DCSpatial, [121](#)
- * **'subject'**
 - DCSubject, [122](#)
- * **'tableOfContents'**
 - DCTableOfContents, [123](#)
- * **'temporal'**
 - DCTemporal, [124](#)
- * **'title'**
 - DCTitle, [125](#)
- * **'type'**
 - DCType, [126](#)
- * **'valid'**
 - DCValid, [127](#)
- * **API**
 - SwordClient, [134](#)
 - SwordDataverseClient, [136](#)
 - SwordHalClient, [140](#)
- * **Atom**
 - AtomAuthor, [11](#)
 - AtomContributor, [14](#)
 - AtomPerson, [31](#)
 - AtomPubClient, [33](#)
 - SwordServiceDocument, [142](#)
- * **Author**
 - AtomAuthor, [11](#)
 - AtomContributor, [14](#)
- * **Category**
 - AtomCategory, [12](#)
- * **Client**
 - SwordClient, [134](#)
 - SwordDataverseClient, [136](#)
 - SwordHalClient, [140](#)
- * **Core**
 - DCAbstract, [36](#)
 - DCAccessRights, [37](#)
 - DCAccrualMethod, [38](#)
 - DCAccrualPeriodicity, [39](#)
 - DCAccrualPolicy, [40](#)
 - DCAlternative, [41](#)
 - DCAudience, [42](#)
 - DCAvailable, [43](#)
 - DCBibliographicCitation, [44](#)
 - DCConformsTo, [45](#)
 - DCCContributor, [46](#)
 - DCCoverage, [47](#)
 - DCCreated, [48](#)
 - DCCreator, [49](#)
 - DCDate, [50](#)
 - DCDateAccepted, [51](#)
 - DCDateCopyrighted, [52](#)
 - DCDateSubmitted, [53](#)
 - DCDescription, [54](#)
 - DCEducationalLevel, [55](#)
 - DCElement, [56](#)
 - DCEntry, [57](#)
 - DCExtent, [94](#)
 - DCFormat, [95](#)
 - DCHasPart, [96](#)
 - DCHasVersion, [97](#)
 - DCIdentifier, [98](#)
 - DCInstructionalMethod, [99](#)
 - DCIsPartOf, [100](#)
 - DCIsReferencedBy, [101](#)
 - DCIsReplacedBy, [102](#)
 - DCIsRequiredBy, [103](#)
 - DCIssued, [104](#)
 - DCIsVersionOf, [105](#)
 - DCLanguage, [106](#)
 - DCLicense, [107](#)
 - DCMediator, [108](#)
 - DCMedium, [109](#)
 - DCMIVocabulary, [110](#)
 - DCModified, [111](#)
 - DCProvenance, [112](#)
 - DCPublisher, [113](#)
 - DCReferences, [114](#)
 - DCRelation, [115](#)
 - DCReplaces, [116](#)
 - DCRequires, [117](#)
 - DCRights, [118](#)
 - DCRightsHolder, [119](#)
 - DCSource, [120](#)

- DCSpatial, [121](#)
- DCSubject, [122](#)
- DCTableOfContents, [123](#)
- DCTemporal, [124](#)
- DCTitle, [125](#)
- DCType, [126](#)
- DCValid, [127](#)
- * **Dataverse**
 - SwordDataverseClient, [136](#)
- * **Dublin**
 - DCAbstract, [36](#)
 - DCAccessRights, [37](#)
 - DCAccrualMethod, [38](#)
 - DCAccrualPeriodicity, [39](#)
 - DCAccrualPolicy, [40](#)
 - DCAAlternative, [41](#)
 - DCAudience, [42](#)
 - DCAvailable, [43](#)
 - DCBibliographicCitation, [44](#)
 - DCConformsTo, [45](#)
 - DCCContributor, [46](#)
 - DCCoverage, [47](#)
 - DCCreated, [48](#)
 - DCCreator, [49](#)
 - DCDate, [50](#)
 - DCDateAccepted, [51](#)
 - DCDateCopyrighted, [52](#)
 - DCDateSubmitted, [53](#)
 - DCDescription, [54](#)
 - DCEducationalLevel, [55](#)
 - DCElement, [56](#)
 - DCEntry, [57](#)
 - DCExtent, [94](#)
 - DCFormat, [95](#)
 - DCHasPart, [96](#)
 - DCHasVersion, [97](#)
 - DCIdentifier, [98](#)
 - DCInstructionalMethod, [99](#)
 - DCIsPartOf, [100](#)
 - DCIsReferencedBy, [101](#)
 - DCIsReplacedBy, [102](#)
 - DCIsRequiredBy, [103](#)
 - DCIssued, [104](#)
 - DCIsVersionOf, [105](#)
 - DCLanguage, [106](#)
 - DCLicense, [107](#)
 - DCMediator, [108](#)
 - DCMedium, [109](#)
 - DCMIVocabulary, [110](#)
 - DCModified, [111](#)
 - DCProvenance, [112](#)
 - DCPublisher, [113](#)
 - DCReferences, [114](#)
 - DCRelation, [115](#)
 - DCReplaces, [116](#)
 - DCRequires, [117](#)
 - DCRights, [118](#)
 - DCRightsHolder, [119](#)
 - DCSource, [120](#)
 - DCSpatial, [121](#)
 - DCSubject, [122](#)
 - DCTableOfContents, [123](#)
 - DCTemporal, [124](#)
 - DCTitle, [125](#)
 - DCType, [126](#)
 - DCValid, [127](#)
- * **Entry**
 - AtomEntry, [15](#)
 - DCEntry, [57](#)
- * **ISO**
 - AtomNamespace, [30](#)
- * **Link**
 - AtomLink, [27](#)
- * **Person**
 - AtomPerson, [31](#)
 - AtomPubClient, [33](#)
 - SwordServiceDocument, [142](#)
- * **SWORD**
 - SwordClient, [134](#)
 - SwordDataverseClient, [136](#)
 - SwordHalClient, [140](#)
- * **atom**
 - AtomAbstractObject, [6](#)
 - AtomCategory, [12](#)
 - AtomEntry, [15](#)
 - AtomFeed, [20](#)
 - AtomLink, [27](#)
- * **dc**
 - DCEntry, [57](#)
- * **element**
 - DCAbstract, [36](#)
 - DCAccessRights, [37](#)
 - DCAccrualMethod, [38](#)
 - DCAccrualPeriodicity, [39](#)
 - DCAccrualPolicy, [40](#)
 - DCAAlternative, [41](#)

- DCAudience, [42](#)
- DCAvailable, [43](#)
- DCBibliographicCitation, [44](#)
- DCConformsTo, [45](#)
- DCCContributor, [46](#)
- DCCoverage, [47](#)
- DCCreated, [48](#)
- DCCreator, [49](#)
- DCDate, [50](#)
- DCDateAccepted, [51](#)
- DCDateCopyrighted, [52](#)
- DCDateSubmitted, [53](#)
- DCDescription, [54](#)
- DCEducationalLevel, [55](#)
- DCElement, [56](#)
- DCExtent, [94](#)
- DCFormat, [95](#)
- DCHasPart, [96](#)
- DCHasVersion, [97](#)
- DCIdentifier, [98](#)
- DCInstructionalMethod, [99](#)
- DCIsPartOf, [100](#)
- DCIsReferencedBy, [101](#)
- DCIsReplacedBy, [102](#)
- DCIsRequiredBy, [103](#)
- DCIssued, [104](#)
- DCIsVersionOf, [105](#)
- DCLanguage, [106](#)
- DCLicense, [107](#)
- DCMediator, [108](#)
- DCMedium, [109](#)
- DCMIVocabulary, [110](#)
- DCModified, [111](#)
- DCProvenance, [112](#)
- DCPublisher, [113](#)
- DCReferences, [114](#)
- DCRelation, [115](#)
- DCReplaces, [116](#)
- DCRequires, [117](#)
- DCRights, [118](#)
- DCRightsHolder, [119](#)
- DCSource, [120](#)
- DCSpatial, [121](#)
- DCSubject, [122](#)
- DCTableOfContents, [123](#)
- DCTemporal, [124](#)
- DCTitle, [125](#)
- DCType, [126](#)
- DCValid, [127](#)
- * **feed**
 - AtomFeed, [20](#)
- * **logger**
 - atom4RLogger, [4](#)
- * **metadata**
 - AtomNamespace, [30](#)
- * **namespace**
 - AtomNamespace, [30](#)
- atom4R, [3](#)
- atom4R-package (atom4R), [3](#)
- atom4R::atom4RLogger, [6, 11, 12, 14, 15, 21, 28, 31, 33, 36–57, 94–109, 111–127, 134, 136, 140, 142](#)
- atom4R::AtomAbstractObject, [11, 12, 14, 15, 21, 28, 31, 36–57, 94–109, 111–127](#)
- atom4R::AtomEntry, [57](#)
- atom4R::AtomPerson, [11, 14](#)
- atom4R::AtomPubClient, [134, 136, 140](#)
- atom4R::DCAudience, [55, 108](#)
- atom4R::DCCoverage, [121, 124](#)
- atom4R::DCDate, [43, 48, 51–53, 104, 111, 127](#)
- atom4R::DCDescription, [36, 123](#)
- atom4R::DCElement, [36–55, 94–109, 111–127](#)
- atom4R::DCFormat, [94, 109](#)
- atom4R::DCIdentifier, [44](#)
- atom4R::DCRelation, [45, 114, 116, 117, 120](#)
- atom4R::DCRights, [37, 107](#)
- atom4R::DCTitle, [41](#)
- atom4R::SwordClient, [136, 140](#)
- atom4RLogger, [4](#)
- AtomAbstractObject, [6, 7, 128](#)
- AtomAuthor, [11, 11, 17, 18, 24](#)
- AtomCategory, [12, 13](#)
- AtomContributor, [14, 14, 18, 24, 25](#)
- AtomEntry, [15, 16, 26, 138, 139](#)
- AtomFeed, [20, 22, 137, 138, 141](#)
- AtomLink, [27, 28](#)
- AtomNamespace, [30, 30](#)
- AtomPerson, [31, 31](#)
- AtomPubClient, [33, 35, 137](#)
- character, [63–92](#)
- Date, [68, 71–73, 81, 84](#)
- DCAbstract, [36, 36, 63, 64](#)

- DCAccessRights, [37, 37, 64](#)
- DCAccrualMethod, [38, 38, 65](#)
- DCAccrualPeriodicity, [39, 39, 65, 66](#)
- DCAccrualPolicy, [40, 40, 66](#)
- DCAlternative, [41, 41, 66, 67](#)
- DCAudience, [42, 42, 67](#)
- DCAvailable, [43, 43, 68](#)
- DCBibliographicCitation, [44, 44, 68, 69](#)
- DCConformsTo, [45, 45, 69](#)
- DCContributor, [46, 69, 70](#)
- DCCoverage, [47, 70](#)
- DCCreated, [48, 48, 71](#)
- DCCreator, [49, 71](#)
- DCDate, [50, 72](#)
- DCDateAccepted, [51, 51, 72](#)
- DCDateCopyrighted, [52, 52, 73](#)
- DCDateSubmitted, [53, 53, 73](#)
- DCDescription, [54, 73, 74](#)
- DCEducationalLevel, [55, 55, 74](#)
- DCElement, [56, 56, 63](#)
- DCEntry, [57, 62](#)
- DCExtent, [75, 94, 94](#)
- DCFormat, [75, 76, 95, 95](#)
- DCHasPart, [76, 96](#)
- DCHasVersion, [76, 77, 97](#)
- DCIdentifier, [77, 98](#)
- DCInstructionalMethod, [78, 99, 99](#)
- DCIsPartOf, [78, 79, 100](#)
- DCIsReferencedBy, [79, 101](#)
- DCIsReplacedBy, [79, 80, 102](#)
- DCIsRequiredBy, [80, 103](#)
- DCIssued, [81, 104, 104](#)
- DCIsVersionOf, [81, 105](#)
- DCLanguage, [81, 82, 106](#)
- DCLicense, [82, 107, 107](#)
- DCMediator, [83, 108, 108](#)
- DCMedium, [83, 84, 109, 109](#)
- DCMIVocabulary, [110](#)
- DCModified, [84, 111, 111](#)
- DCProvenance, [84, 85, 112, 112](#)
- DCPublisher, [85, 113](#)
- DCReferences, [85, 86, 114, 114](#)
- DCRelation, [86, 115](#)
- DCReplaces, [87, 116, 116](#)
- DCRequires, [87, 88, 117, 117](#)
- DCRights, [88, 118](#)
- DCRightsHolder, [88, 89, 119, 119](#)
- DCSource, [89, 120](#)
- DCSpatial, [121, 121](#)
- DCSubject, [90, 122](#)
- DCTableOfContents, [90, 91, 123, 123](#)
- DCTemporal, [91, 124, 124](#)
- DCTitle, [91, 92, 125](#)
- DCType, [92, 126](#)
- DCValid, [127, 127](#)
- getAtomClasses, [128](#)
- getAtomNamespace, [128](#)
- getAtomNamespaces, [129](#)
- getAtomSchemas, [129](#)
- getClassesInheriting, [130](#)
- getDCMIVocabularies, [130](#)
- getDCMIVocabulary, [131](#)
- POSIXt, [68, 71–73, 81, 84](#)
- R6Class, [4–6, 10, 11, 14, 15, 20, 28, 30, 31, 33, 36–57, 94–127, 134, 136, 140, 142](#)
- readDCEntry, [131](#)
- registerAtomNamespace, [132](#)
- registerAtomSchema, [133](#)
- setAtomNamespaces, [133](#)
- setAtomSchemas, [133](#)
- setDCMIVocabularies, [134](#)
- SwordClient, [134, 135](#)
- SwordDataverseClient, [136](#)
- SwordHalClient, [140](#)
- SwordServiceDocument, [35, 135, 137, 141, 142, 142](#)
- XMLInternalNode-class, [7, 9, 11, 13, 14, 16, 22, 28, 32, 36–56, 62, 94–109, 111–127, 142](#)