Package: arcticdatautils (via r-universe)

September 7, 2024

Title Utilities for the Arctic Data Center

Version 0.7.0

Description A set of utilities for working with the Arctic Data Center (https://arcticdata.io).

License Apache License (== 2.0)

URL https://nceas.github.io/arcticdatautils/

BugReports https://github.com/NCEAS/arcticdatautils/issues

Encoding UTF-8

LazyData true

Depends R (>= 3.2.3)

Imports dataone, datapack, dplyr, digest, EML (>= 2.0), httr, jsonlite (>= 1.8.4), magrittr, methods, filelock, stringr, stringi, tools, uuid, xml2, XML, lifecycle, rdflib, pins, rlang

Suggests emld, humaniformat, knitr, lubridate, ncdf4, RCurl, purrr, raster, rmarkdown, sf, testthat (>= 3.0.0), xslt, yaml

RoxygenNote 7.2.3

Roxygen list(markdown = TRUE)

VignetteBuilder knitr

Config/testthat/edition 3

Repository https://dataoneorg.r-universe.dev

RemoteUrl https://github.com/NCEAS/arcticdatautils

RemoteRef HEAD

RemoteSha 62c4df83d9128746e2ed1d14383bab6856861e87

Contents

`arcticdatautils` *arcticdatautils: Utilities for the Arctic Data Center*

Description

This package contains code for doing lots of useful stuff that's too specific for the dataone package, primarily functions that streamline Arctic Data Center operations.

convert_iso_to_eml *Convert an ISO document to EML using an XSLT*

Description

Leave style=NA if you want to use the default ISO-to-EML stylesheet.

Usage

```
convert_iso_to_eml(path, style = NA)
```

Arguments

path	(character) Path to the file to convert.
style	(xslt) The XSLT object to be used for transformation.

Value

(character) Location of the converted file.

Examples

```
## Not run:
iso_path <- "~/Docuements/ISO_metadata.xml"
eml_path <- convert_iso_to_eml(iso_path)

## End(Not run)
```

create_dummy_attributes_dataframe *Create test attributes data.frame*

Description

Create a test data.frame of attributes.

Usage

```
create_dummy_attributes_dataframe(numberAttributes, factors = NULL)
```

Arguments

numberAttributes	(integer) Number of attributes to be created in the table.
factors	(character) Optional vector of factor names to include.

Value

(data.frame) A data.frame of attributes.

Examples

```
## Not run:
# Create dummy attribute dataframe with 6 attributes and 1 factor
attributes <- create_dummy_attributes_dataframe(6, c("Factor1", "Factor2"))

## End(Not run)
```

`create_dummy_enumeratedDomain_dataframe` *Create test enumeratedDomain data.frame*

Description

Create a test data.frame of enumeratedDomains.

Usage

```
create_dummy_enumeratedDomain_dataframe(factors)
```

Arguments

`factors`	(character) Vector of factor names to include.

Value

(data.frame) A data.frame of factors.

Examples

```
## Not run:
# Create dummy dataframe of 2 factors/enumerated domains
attributes <- create_dummy_enumeratedDomain_dataframe(c("Factor1", "Factor2"))

## End(Not run)
```

create_dummy_metadata *Create a test metadata object*

Description

Create a test EML metadata object.

Usage

```
create_dummy_metadata(mn, data_pids = NULL)
```

Arguments

mn	(MNode) The Member Node.
data_pids	(character) Optional. PIDs for data objects the metadata documents.

Value

(character) The PID of the published metadata document.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pid <- create_dummy_metadata(mn)

## End(Not run)
```

create_dummy_object *Create a test object*

Description

Create a test data object. Make sure the member node you use is not a production node.

Usage

```
create_dummy_object(mn)
```

Arguments

mn	(MNode) The Member Node.

Value

(character) The PID of the dummy object.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")

pid <- create_dummy_object(mn)

## End(Not run)
```

`create_dummy_package` *Create a test package*

Description

Create a full test data package with data objects and 1 metadata object. Size = the number of data objects you want in the dummy package + 1 metadata object.

Usage

```
create_dummy_package(mn, size = 2)
```

Arguments

`mn`	(MNode) The Member Node.
`size`	(numeric) The number of files in the package, including the metadata file.

Value

(list) The PIDs for all elements in the data package.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
#Create dummy package with 5 data objects and 1 metadata object
pids <- create_dummy_package(mn, 6)

## End(Not run)
```

create_dummy_package_full

Create dummy package with fuller metadata

Description

Creates a more complete package than `create_dummy_package()` but is otherwise based on the same concept. This dummy package includes multiple data objects, responsible parties, geographic locations, method steps, etc.

Usage

```
create_dummy_package_full(mn, title = "A Dummy Package")
```

Arguments

mn	(MNode) The Member Node.
title	(character) Optional. Title of package. Defaults to "A Dummy Package".

Value

(list) The PIDs for all elements in the data package.

create_dummy_parent_package

Create a test parent package

Description

Create a test parent data package. Make sure the node is not a production node.

Usage

```
create_dummy_parent_package(mn, children)
```

Arguments

mn	(MNode) The Member Node.
children	(character) Child package (resource maps) PIDs.

Value

(list) The resource map PIDs for both the parent and child packages.

Examples

```
## Not run:
# Set environment

## End(Not run)
```

`create_resource_map` *Create a resource map object on a Member Node*

Description

This function first generates a new resource map RDF/XML document locally and then uses the `dataone::createObject()` function to create the object on the specified MN.

Usage

```
create_resource_map(
  mn,
  metadata_pid,
  data_pids = NULL,
  child_pids = NULL,
  check_first = TRUE,
  ...
)
```

Arguments

`mn`	(MNode) The Member Node
`metadata_pid`	(character) The PID of the metadata object to go in the package.
`data_pids`	(character) The PID(s) of the data objects to go in the package.
`child_pids`	(character) The resource map PIDs of the packages to be nested under the package.
`check_first`	(logical) Optional. Whether to check the PIDs passed in as arguments exist on the MN before continuing. This speeds up the function, especially when `data_pids` has many elements.
`...`	Additional arguments that can be passed into `publish_object()`.

Details

If you only want to generate resource map RDF/XML, see `generate_resource_map()`.

Value

(character) The PID of the created resource map.

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")

meta_pid <- 'urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe'
dat_pid <- c('urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1',
'urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe')

create_resource_map(mn, metadata_pid = meta_pid, data_pids = dat_pid)

## End(Not run)
```

---

eml_adcad_annotation *Given a term from the ADC Academic Disciplines (ADCAD) ontology, produce the corresponding annotation*

Description

Reduces the amount of copy pasting needed

Usage

```
eml_adcad_annotation(valueLabel)
```

Arguments

valueLabel	(character) One of the disciplines found in ADCAD

Value

list - a formatted EML annotation

Examples

```
eml_ecso_annotation("latitude coordinate")
```

eml_add_distribution *Add distribution information to EML*

Description

Adds a landing page URL to the dataset, and corrects the metadata identifier by replacing the existing identifier with that which is passed. Note that this function constructs landing page URLs for the Arctic Data Center only and will not work correctly on other repositories.

Usage

```
eml_add_distribution(doc, identifier)
```

Arguments

doc	(emld) An EML document
identifier	(character) A pre-issued, unassigned identifier (as from dataone::generateIdentifier())

Value

doc (emld) An EML document with distribution added

Examples

```
## Not run:
library(EML)
d1c <- dataone::D1Client("STAGING", "mnTestARCTIC")
# read in any EML document
doc <- read_eml(system.file("extdata/strix-pacific-northwest.xml", package="dataone"))
# generate a doi
id <- generateIdentifier(d1c@mn, "doi")
doc <- eml_add_distribution(doc, id)

## End(Not run)
```

eml_add_entity_system *Add system information to entities*

Description

This function adds system information to entities in a document

Usage

```
eml_add_entity_system(doc)
```

Arguments

doc	(emld) An EML document

Value

(emld) An EML document

Examples

```
## Not run:
# Add publisher information to an existing document
doc <- eml_add_entity_system(doc)

## End(Not run)
```

---

eml_add_publisher *Add publisher information to EML document*

Description

This function adds Arctic Data Center publisher information to an EML document

Usage

```
eml_add_publisher(doc)
```

Arguments

doc	(emld) An EML document

Value

(emld) An EML document

Examples

```
## Not run:
# Add publisher information to an existing document
doc <- eml_add_publisher(doc)

## End(Not run)
```

eml_arcrc_add_annotation

Add an Arctic Report Card annotation to a dataset

Description

Creates an annotation from the Arctic Report Card ontology here and inserts the annotation into the EML document doc while retaining any existing annotations such as the sensitivity annotations or dataset categorization. For a list of available essay topics or key variables, see link above.

Usage

```
eml_arcrc_add_annotation(doc, property, label)
```

Arguments

doc	(emld) An EML document
property	(character) One of two properties: "isAbout" for key variables or "influenced" for essay topics
label	(character) One or more labels in title case from the ADCAD ontology.

Value

doc (emld) An EML document with annotation added

Examples

```
library(EML)
# read in any EML document
doc <- read_eml(system.file("extdata/strix-pacific-northwest.xml", package="dataone"))
# add the dataset categories
doc <- eml_arcrc_add_annotation(doc, "isAbout", c("sea ice thickness", "sea surface temperature"))
```

eml_arcrc_essay_annotation

Given an essay topic from the Arctic Report Card (ARCRC) ontology, produce the corresponding annotation

Description

Reduces the amount of copy pasting needed

Usage

```
eml_arcrc_essay_annotation(valueLabel)
```

Arguments

`valueLabel` (character) One of the essay topics found in ARCRC

Value

list - a formatted EML annotation

Examples

```
eml_arcrc_essay_annotation("Sea Ice Indicator")
```

---

eml_arcrc_key_variable_annotation

Given a key variable from the Arctic Report Card (ARCRC) ontology, produce the corresponding annotation

---

Description

Reduces the amount of copy pasting needed

Usage

```
eml_arcrc_key_variable_annotation(valueLabel)
```

Arguments

`valueLabel` (character) One of the key variables found in ARCRC

Value

list - a formatted EML annotation

Examples

```
eml_arcrc_key_variable_annotation("age of sea ice")
```

eml_associated_party *Create an EML associatedParty*

Description

See `eml_party()` for details.

Usage

```
eml_associated_party(...)
```

Arguments

`...`	Arguments passed on to `eml_party()`.

Value

(associatedParty) The new associatedParty.

Examples

```
## Not run:
eml_associated_party("test", "user", email = "test@user.com", role = "Principal Investigator")

## End(Not run)
```

eml_categorize_dataset *Categorize a dataset with an annotation*

Description

Creates an annotation from the ADC Academic Disciplines ontology here and inserts the annotation into the EML document doc while retaining any existing annotations such as the sensitivity annotations. For a list of available disciplines, see link above.

Usage

```
eml_categorize_dataset(doc, discipline)
```

Arguments

doc	(emld) An EML document
discipline	(character) One or more disciplines in title case from the ADCAD ontology.

Value

doc (emld) An EML document with annotation added

Examples

```
library(EML)
# read in any EML document
doc <- read_eml(system.file("extdata/strix-pacific-northwest.xml", package="dataone"))
# add the dataset categories
doc <- eml_categorize_dataset(doc, c("Soil Science", "Ecology"))
```

`eml_contact` *Create an EML contact. Contact information is passed on to* `eml_party()`

Description

[Deprecated]

Usage

```
eml_contact(...)
```

Arguments

`...`	Arguments passed on to `eml_party()`.

Details

Please use the constructors in the EML package instead

Value

(contact) The new contact.

Examples

```
## Not run:
eml_contact("test", "user", email = "test@user.com")
eml_creator("creator", "Bryce", "Mecum", userId = "https://orcid.org/0000-0002-0381-3766")
eml_creator("creator", c("Dominic", "'Dom'"), "Mullen", c("NCEAS", "UCSB"),
            c("Data Scientist", "Programmer"))

## End(Not run)
```

eml_creator *Create an EML creator.*

Description

[Deprecated]

Usage

```
eml_creator(...)
```

Arguments

...	Arguments passed on to eml_party().

Details

Please use the constructors in the EML package instead

Value

(creator) The new creator.

Examples

```
## Not run:
eml_creator("test", "user", email = "test@user.com")
eml_creator("creator", "Bryce", "Mecum", userId = "https://orcid.org/0000-0002-0381-3766")
eml_creator("creator", c("Dominic", "'Dom'"), "Mullen", c("NCEAS", "UCSB"),
            c("Data Scientist", "Programmer"))

## End(Not run)
```

eml_ecso_annotation *Given a an annotation from the ECSO ontology, produce the corresponding annotation*

Description

Reduces the amount of copy pasting needed

Usage

```
eml_ecso_annotation(valueLabel)
```

Arguments

valueLabel	(character) the label for the annotation found in ECSO

Value

list - a formatted EML annotation

Examples

```
eml_ecso_annotation("latitude coordinate")
```

---

eml_get_raster_metadata

Get raster info from a file on disk

Description

This function populates a spatialRaster element with the required elements by reading a local raster file in. The coord_name argument can be found by examining the data.frame that get_coord_list() returns against the proj4string of the raster file.

Usage

```
eml_get_raster_metadata(path, coord_name = NULL, attributes)
```

Arguments

path	(char) Path to a raster file
coord_name	(char) horizCoordSysDef name
attributes	(dataTable) attributes for raster

---

eml_get_simple

Get a simple list output from EML::eml_get()

Description

This function is a convenience wrapper around EML::eml_get() which returns the output as a simple list as opposed to an object of type emld by removing the attributes and context from the object. If an element containing children is returned all of it's children will be flattened into a named character vector. This function is best used to extract values from elements that have no children.

Usage

```
eml_get_simple(doc, element)
```

Arguments

doc	(list) An EML object or child/descendant object
element	(character) Name of the element to be extracted. If multiple occurrences are found, will extract all.

Value

out (vector) A list of values contained in element given

Examples

```
## Not run:
cn <- dataone::CNode('PROD')
adc <- dataone::getMNode(cn,'urn:node:ARCTIC')

doc <- EML::read_eml(dataone::getObject(adc, 'doi:10.18739/A2S17SS1M'))

datatable_names <- eml_get_simple(doc$dataset$dataTable, element = "entityName")

## End(Not run)
```

---

eml_nsf_to_project *Create an EML project section from a list of NSF award numbers*

Description

This function takes a list of NSF award numbers and uses it to query the NSF API to get the award title, PIs, and coPIs. The return value is an EML project section. The function supports 1 or more award numbers

Usage

```
eml_nsf_to_project(awards, eml_version = "2.2")
```

Arguments

awards	(list) A list of NSF award numbers as characters
eml_version	(char) EML version to use (2.1.1 or 2.2.0)

Value

project (emld) An EML project section

Examples

```
awards <- c("1203146", "1203473", "1603116")

proj <- eml_nsf_to_project(awards, eml_version = "2.1.1")

me <- list(individualName = list(givenName = "Jeanette", surName = "Clark"))

doc <- list(packageId = "id", system = "system",
           dataset = list(title = "A Mimimal Valid EML Dataset",
                          creator = me,
                          contact = me))

doc$dataset$project <- proj

EML::eml_validate(doc)
```

`eml_otherEntity_to_dataTable` *Convert otherEntities to dataTables*

Description

Convert an EML 'otherEntity' object to a 'dataTable' object. This will convert an otherEntity object as currently constructed - it does not add a physical or add attributes. However, if these are already in their respective slots, they will be retained.

Usage

```
eml_otherEntity_to_dataTable(doc, index, validate_eml = TRUE)
```

Arguments

`doc`	(list) An EML document.
`index`	(integer) The indicies of the otherEntities to be transformed.
`validate_eml`	(logical) Optional. Whether or not to validate the EML after completion. Setting this to `FALSE` reduces execution time by ~50 percent.

Author(s)

Dominic Mullen dmullen17@gmail.com

Examples

```
## Not run:
doc <- read_eml(system.file("example-eml.xml", package = "arcticdatautils"))

doc <- eml_otherEntity_to_dataTable(doc, 1)

## End(Not run)
```

`eml_party` *Create an EML party*

Description

[Deprecated]

Usage

```
eml_party(
  type = "associatedParty",
  given_names = NULL,
  sur_name = NULL,
  organization = NULL,
  position = NULL,
  email = NULL,
  phone = NULL,
  address = NULL,
  userId = NULL,
  role = NULL
)
```

Arguments

`type`	(character) The type of party (e.g. 'contact').
`given_names`	(character) The party's given name(s).
`sur_name`	(character) The party's surname.
`organization`	(character) The party's organization name.
`position`	(character) The party's position.
`email`	(character) The party's email address(es).
`phone`	(character) The party's phone number(s).
`address`	(character) The party's address(es) as a valid EML address
`userId`	(character) The party's ORCID, in format https://orcid.org/WWWW-XXXX-YYYY-ZZZZ.
`role`	(character) The party's role.

Details

Please use the constructors in the EML package instead

You will usually want to use the high-level functions such as `eml_creator()` and `eml_contact()` but using this is fine.

The `userId` argument assumes an ORCID so be sure to adjust for that.

Value

(party) An instance of the party specified by the type argument.

Examples

```
## Not run:
eml_party("creator", "Test", "User")
eml_party("creator", "Bryce", "Mecum", userId = "https://orcid.org/0000-0002-0381-3766")
eml_party("creator", given_names = list("Dominic", "'Dom'"),
                     sur_name = "Mullen", list("NCEAS", "UCSB"),
                     position = list("Data Scientist", "Programmer"),
                         address = eml$address(deliveryPoint = "735 State St",
                         city = "Santa Barbara",
                         administrativeArea = "CA",
                         postalCode = "85719"))

## End(Not run)
```

---

eml_set_reference *Set a reference to an EML object*

Description

[Deprecated]

Usage

```
eml_set_reference(element_to_reference, element_to_replace)
```

Arguments

`element_to_reference`
: (list) An EML element to reference.

`element_to_replace`
: (list) An EML element to replace with a reference.

Details

please add references directly instead

This function creates a new object with the same class as `element_to_replace` using a reference to `element_to_reference`.

Author(s)

Dominic Mullen dmullen17@gmail.com

Examples

```
## Not run:
cn <- dataone::CNode('PROD')
adc <- dataone::getMNode(cn,'urn:node:ARCTIC')
doc <- EML::read_eml(dataone::getObject(adc, 'doi:10.18739/A2S17SS1M'))

# Set the first contact as a reference to the first creator
doc$dataset$contact[[1]] <- eml_set_reference(doc$dataset$creator[[1]],
doc$dataset$contact[[1]])

# This is also useful when we want to set references to a subset of 'dataTable'
  or 'otherEntity' objects
# Add a few more objects first to illustrate the use:
doc$dataset$dataTable[[3]] <- doc$dataset$dataTable[[1]]
doc$dataset$dataTable[[4]] <- doc$dataset$dataTable[[1]]
# Add references to the second and third elements only (not the 4th):
for (i in 2:3) {
    doc$dataset$dataTable[[i]]$attributeList <- eml_set_reference(
                                                  doc$dataset$dataTable[[1]]$attributeList,
                                                  doc$dataset$dataTable[[i]]$attributeList)
}
# If we print the entire 'dataTable' list we see elements 2 and 3 have
references while 4 does not.

doc$dataset$dataTable

## End(Not run)
```

`eml_set_shared_attributes`
Set shared attribute references

Description

[Deprecated]

Usage

```
eml_set_shared_attributes(doc, attributeList = NULL, type = "dataTable")
```

Arguments

`doc`	(emld) An EML object.
`attributeList`	(attributeList) Optional. An EML attributeList object. If not provided then it will default to the attributeList of the first `type` element.
`type`	(character) Optional. Specifies whether to replace 'dataTable' or 'otherEntity' attributeList objects with references. Defaults to 'dataTable'.

Details

please add references directly instead

This function sets shared attributes using the attributes of the first `type` selected and creates references for all remaining objects of equivalent `type`.

Value

(doc) The modified EML document.

Author(s)

Dominic Mullen dmullen17@gmail.com

Examples

```
## Not run:
cn <- dataone::CNode('PROD')
adc <- dataone::getMNode(cn,'urn:node:ARCTIC')
doc <- EML::read_eml(dataone::getObject(adc, 'doi:10.18739/A2S17SS1M'))
atts <- EML::set_attributes(
               EML::get_attributes(eml$dataset$dataTable[[1]]$attributeList)$attributes)

eml <- eml_set_shared_attributes(eml, atts, type = 'dataTable')

## End(Not run)
```

---

env_get *Get the current environment name*

Description

Get the current environment name.

Usage

```
env_get()
```

Value

(character) The environment name.

find_newest_object *Find the newest object within the given set of objects*

Description

Find the newest object, based on dateUploaded, within the given set of objects.

Usage

```
find_newest_object(node, identifiers, rows = 1000)
```

Arguments

node	(MNode/CNode) The Member Node to query.
identifiers	(character) One or more identifiers.
rows	(numeric) Optional. Specify the size of the query result set.

Value

(character) The PID of the newest object. In the case of a tie (very unlikely) the first element, in natural order, is returned.

Examples

```
## Not run:
mn <- MNode(...)
find_newest_object(mn, c("PIDX", "PIDY", "PIDZ"))

## End(Not run)
```

format_eml *Generate the EML 2.1.1 format ID*

Description

Returns the EML 2.1.1 format ID.

Usage

```
format_eml(version)
```

Arguments

version	The version of EML ('2.1.1' or '2.2.0')

Value

(character) The format ID for EML 2.1.1.

Examples

```
format_eml("2.1.1")
## Not run:
# Upload a local EML 2.1.1 file:
env <- env_load()
publish_object(env$mn, "path_to_some_EML_file", format_eml("2.1"))

## End(Not run)
```

---

format_iso *Generate the ISO 19139 format ID*

Description

Returns the ISO 19139 format ID.

Usage

```
format_iso()
```

Value

(character) The format ID for ISO 19139.

Examples

```
format_iso()
## Not run:
# Upload a local ISO19139 XML file:
env <- env_load()
publish_object(env$mn, "path_to_some_EML_file", format_iso())

## End(Not run)
```

generate_resource_map *Create a resource map RDF/XML file and save is to a temporary path*

Description

This is a convenience wrapper around the constructor of the ResourceMap class from DataPackage.

Usage

```
generate_resource_map(
  metadata_pid,
  data_pids = NULL,
  child_pids = NULL,
  other_statements = NULL,
  resolve_base = "https://cn.dataone.org/cn/v2/resolve",
  resource_map_pid = NULL
)
```

Arguments

metadata_pid	(character) PID of the metadata object.
data_pids	(character) PID(s) of the data objects.
child_pids	(character) Optional. PID(s) of child resource maps.
other_statements	(data.frame) Extra statements to add to the resource map.
resolve_base	(character) Optional. The resolve service base URL.
resource_map_pid	(character) The PID of a resource map.

Value

(character) Absolute path to the resource map on disk.

Examples

```
## Not run:
generate_resource_map("X", "Y", "Z",
                      other_statements = data.frame(subject="http://example.com/me",
                                                    predicate="http://example.com/foo",
                                                    object="http://example.com/bar"))

## End(Not run)
```

get_all_versions *Get the PIDs of all versions of an object*

Description

Get the PIDs of all versions of an object.

Usage

```
get_all_versions(node, pid)
```

Arguments

node	(MNode) The Member Node to query.
pid	(character) Any object in the chain.

Value

(character) A vector of PIDs in the chain, in order.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pid <- "urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1"

ids <- get_all_versions(mn, pid)

## End(Not run)
```

get_coord_list *Get list of Coordinate Reference Systems*

Description

Get a data.frame of EML coordinate reference systems that can be searched and filtered more easily than the raw XML file.

Usage

```
get_coord_list()
```

get_mn_base_url *Get base URL of a Member Node*

Description

Get the base URL of a Member Node.

Usage

```
get_mn_base_url(mn)
```

Arguments

mn	(character) The Member Node.

Value

(character) The URL.

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")

## End(Not run)
```

get_ncdf4_attributes *Get a data.frame of attributes from a NetCDF object*

Description

Get a data.frame of attributes from a NetCDF object.

Usage

```
get_ncdf4_attributes(nc)
```

Arguments

nc	(ncdf4/character) Either a ncdf4 object or a file path.

Value

(data.frame) A data.frame of the attributes.

Examples

```
## Not run:
get_ncdf4_attributes("./path/to/my.nc")

## End(Not run)
```

`get_ontology_concepts` *Gets all the concepts*

Description

Takes an ontology and returns a dataframe with all the URIs and labels. This is mainly used for MOSAiC because the ontology is modeled differently

Usage

```
get_ontology_concepts(ontology)
```

Arguments

`ontology`	(list) the list form of a OWL file

Value

dataframe

Examples

```
mosaic <- read_ontology("mosaic")
get_ontology_concepts(mosaic)
```

`get_package` *Get a structured list of PIDs for the objects in a package*

Description

[Deprecated]

Usage

```
get_package(node, pid, file_names = FALSE, rows = 5000)
```

Arguments

node	(MNode/CNode) The Coordinating/Member Node to run the query on.
pid	(character) The the resource map PID of the package.
file_names	(logical) Whether to return file names for all objects.
rows	(numeric) The number of rows to return in the query. This is only useful to set if you are warned about the result set being truncated. Defaults to 5000.

Details

Please use dataone::getDataPackage() when possible

Get a structured list of PIDs for the objects in a package, including the resource map, metadata, and data objects.

Value

(list) A structured list of the members of the package.

Examples

```
## Not run:
#Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pid <- "resource_map_urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1"

ids <- get_package(mn, pid)

## End(Not run)
```

---

get_token *Get the currently set authentication token*

Description

Get the currently set authentication token.

Usage

```
get_token(node)
```

Arguments

node	(MNode/CNode) The Member/Coordinating Node to query.

Value

(character) The token.

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")
get_token(mn)

## End(Not run)
```

`guess_format_id` *Guess format from filename*

Description

Guess format from filename for a vector of filenames.

Usage

```
guess_format_id(filenames)
```

Arguments

`filenames`	(character) A vector of filenames.

Value

(character) DataONE format IDs.

Examples

```
formatid <- guess_format_id("temperature_data.csv")
```

`is_authorized` *Check if user has authorization to perform an action on an object*

Description

Check if the user has authorization to perform an action on an object.

Usage

```
is_authorized(node, ids, action)
```

Arguments

`node`	(MNode/CNode) The Member/Coordinating Node to query.
`ids`	(character) The PID or SID to check.
`action`	(character) One of read, write, or changePermission.

Value

(logical)

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
is_authorized(mn, pids, "write")

## End(Not run)
```

---

is_obsolete *Test whether the object is obsoleted by another object*

Description

Test whether the object is obsoleted by another object

Usage

```
is_obsolete(node, pids)
```

Arguments

node	(MNode|CNode) The Coordinating/Member Node to run the query on.
pids	(character) One or more PIDs to query against.

Value

(logical) Whether or not the object is obsoleted by another object.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pid <- "urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1"

is_obsolete(mn, pid)

## End(Not run)
```

is_public_read *Check whether an object has public read access*

Description

Check whether objects have public read access. No token needs to be set to use this function.

Usage

```
is_public_read(mn, pids, use.names = TRUE)
```

Arguments

mn	(MNode) The Member Node.
pids	(character) The PIDs of the objects to check for public read access.
use.names	(logical) If TRUE, PIDs will be used as names for the result unless PIDs have names already, in which case those names will be used for the result.

Value

(logical) Whether an object has public read access.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
    "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
is_public_read(mn, pids)

## End(Not run)
```

is_token_expired *Determine whether token is expired*

Description

Determine whether the set token is expired.

Usage

```
is_token_expired(node)
```

Arguments

node	(character) The Member Node.

Value

(logical)

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")
is_token_expired(mn)

## End(Not run)
```

`is_token_set` *Test whether a token is set*

Description

Test whether a token is set.

Usage

```
is_token_set(node)
```

Arguments

node	(MNode/CNode) The Member/Coordinating Node to query.

Value

(logical)

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")
is_token_set(mn)

## End(Not run)
```

mdq_run *Score a metadata document against a MetaDIG suite*

Description

This function scores a metadata document against a MetaDIG suite. The default suite is for the Arctic Data Center.

Usage

```
mdq_run(document, suite_id = "arctic.data.center.suite.1")
```

Arguments

document	(eml/character) Either an EML object or path to a file on disk.
suite_id	(character) Specify a suite ID. Should be one of https://quality.nceas.ucsb.edu/quality/suites.

Value

(data.frame) A sorted data.frame of check results.

Examples

```
## Not run:
# Check an EML document you are authoring
library(EML)
mdq_run(new("eml"))

# Check an EML document that is saved to disk
mdq_run(system.file("examples", "example-eml-2.1.1.xml", package = "EML"))

## End(Not run)
```

mosaic_annotate_attribute

Add a MOSAiC (https://mosaic-expedition.org/) attribute annotation (the returned object does not include the id slot)

Description

Add a MOSAiC (https://mosaic-expedition.org/) attribute annotation (the returned object does not include the id slot)

Usage

```
mosaic_annotate_attribute(eventLabel)
```

Arguments

eventLabel	(character) the event ID provided by the researcher

Value

(list) the attribute level annotation

Examples

```
mosaic_annotate_attribute("PS122/2_14-270")
```

---

mosaic_annotate_dataset

Annotating the MOSAiC dataset level annotations

Description

The basis might differ depending on the campaign if it does not follow the pattern PS122/#. This function assumes the use of the Polarstern as the basis. Please verify this field before adding the annotation.

Usage

```
mosaic_annotate_dataset(campaign)
```

Arguments

campaign	(character vector) the campaign number (can be derrived from the eventID), PS122/#

Value

(list) the dataset level annotation

Examples

```
#with one campaign
mosaic_annotate_dataset("PS122/2")

#multiple campaigns
mosaic_annotate_dataset(c("PS122/2", "PS122/1"))
```

mosaic_portal_filter *Creates the choice label pairs to be pasted into a portal document*

Description

The function only selects the annotations that are used for method/devices (there are 500 + options). copy and paste the output into a portal document's choice filters

Usage

```
mosaic_portal_filter(class)
```

Arguments

class	(character) a class in the MOSAiC ontology to get the filters from

Value

character

Examples

```
mosaic_portal_filter("Method/Device")

mosaic_portal_filter("Basis")

mosaic_portal_filter("Campaign")
```

new_uuid *Generate a new UUID PID*

Description

Generate a new UUID PID.

Usage

```
new_uuid()
```

Value

(character) A new UUID PID.

Examples

```
id <- new_uuid()
```

object_exists *Check if an object exists on a Member Node*

Description

This is a simple check for the HTTP status of a /meta/{PID} call on the provided Member Mode.

Usage

```
object_exists(node, pids)
```

Arguments

node	(MNode) The Member Node to query.
pids	(character) The PID(s) to check the existence of.

Value

(logical) Whether the object exists.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")

object_exists(mn, pids)

## End(Not run)
```

parse_resource_map *Parse a resource map into a data.frame*

Description

Parse a resource map into a data.frame.

Usage

```
parse_resource_map(path)
```

Arguments

path	(character) Path to the resource map (an RDF/XML file).

Value

(data.frame) The statements in the resource map.

Examples

```
## Not run:
# Set environment
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")

rm_pid <- "resource_map_urn:uuid:6b2e5753-4a94-4e6f-971c-36420a446ecb"

# Write resource map to file
writeBin(getObject(mn, rm_pid), "~/Documents/resource_map.rdf")
df <- parse_resource_map("~/Documents/resource_map.rdf")

## End(Not run)
```

---

pid_to_eml_entity *Create EML entity with physical section from any DataONE PID*

Description

Create EML entity with physical section from any DataONE PID

Usage

```
pid_to_eml_entity(mn, pid, entity_type = "otherEntity", ...)
```

Arguments

mn	(MNode) Member Node where the PID is associated with an object.
pid	(character) The PID of the object to create the sub-tree for.
entity_type	(character) What kind of object to create from the input. One of "dataTable", "spatialRaster", "spatialVector", "storedProcedure", "view", or "otherEntity".
...	(optional) Additional arguments to be passed to `eml$entityType()`.

Value

(list) The entity object.

Examples

```
## Not run:
# Generate EML otherEntity
pid_to_eml_entity(mn,
                  pid,
                  entity_type = "otherEntity",
                  entityName = "Entity Name",
                  entityDescription = "Description about entity")

## End(Not run)
```

`pid_to_eml_physical` *Create EML physical objects for the given set of PIDs*

Description

This function creates a data object's physical.

Usage

```
pid_to_eml_physical(mn, pid, num_header_lines = 1)
```

Arguments

`mn`	(MNode) Member Node where the PID is associated with an object.
`pid`	(character) The PID of the object to create the physical for.
`num_header_lines`	(double) The number of headers in a csv/Excel file. Default is equal to 1.

Value

(list) A physical object.

Examples

```
## Not run:
# Generate EML physical sections for an object in a data package
phys <- pid_to_eml_physical(mn, pid, num_header_lines)

## End(Not run)
```

publish_object *Publish an object on a Member Node*

Description

Use sensible defaults to publish an object on a Member Node. If identifier is provided, use it, otherwise generate a UUID. If clone_id is provided, then retrieve the system metadata for that identifier and use it to provide rightsHolder, accessPolicy, and replicationPolicy metadata. Note that this function only uploads the object to the Member Node, and does not add it to a data package, which can be done separately.

Usage

```
publish_object(
  mn,
  path,
  format_id = NULL,
  pid = NULL,
  sid = NULL,
  clone_pid = NULL,
  public = TRUE
)
```

Arguments

mn	(MNode) The Member Node to publish the object to.
path	(character) The path to the file to be published.
format_id	(character) Optional. The format ID to set for the object. When not set, guess_format_id() will be used to guess the format ID. Should be a DataONE format ID.
pid	(character) Optional. The PID to use with the object.
sid	(character) Optional. The SID to use with the new object.
clone_pid	(character) PID of object to clone System Metadata from.
public	(logical) Whether object should be given public read access.

Value

pid (character) The PID of the published object.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
my_path <- "/home/Documents/myfile.csv"
pid <- publish_object(mn, path = my_path, format_id = "text/csv", public = FALSE)

## End(Not run)
```

publish_update *Publish an updated data package*

Description

Publish an update to a data package after updating data files or metadata.

Usage

```
publish_update(
  mn,
  metadata_pid,
  resource_map_pid,
  data_pids = NULL,
  child_pids = NULL,
  metadata_path = NULL,
  identifier = NULL,
  use_doi = FALSE,
  parent_resmap_pid = NULL,
  parent_metadata_pid = NULL,
  parent_data_pids = NULL,
  parent_child_pids = NULL,
  public = TRUE,
  check_first = TRUE,
  format_id = NULL
)
```

Arguments

mn	(MNode) The Member Node to update the object on.
metadata_pid	(character) The PID of the EML metadata document to be updated.
resource_map_pid	(character) The PID of the resource map for the package.
data_pids	(character) PID(s) of data objects that will go in the updated package.
child_pids	(character) Optional. Child packages resource map PIDs.
metadata_path	(character or eml) Optional. An eml class object or a path to a metadata file to update with. If this is not set, the existing metadata document will be used.
identifier	(character) Manually specify the identifier for the new metadata object.
use_doi	(logical) Generate and use a DOI as the identifier for the updated metadata object.
parent_resmap_pid	(character) Optional. PID of a parent package to be updated. Not optional if a parent package exists.

parent_metadata_pid
: (character) Optional. Identifier for the metadata document of the parent package. Not optional if a parent package exists.

parent_data_pids
: (character) Optional. Identifier for the data objects of the parent package. Not optional if the parent package contains data objects.

parent_child_pids
: (character) Optional. Resource map identifier(s) of child packages in the parent package. resource_map_pid should not be included. Not optional if the parent package contains other child packages.

public
: (logical) Optional. Make the update public. If FALSE, will set the metadata and resource map to private (but not the data objects). This applies to the new metadata PID and its resource map and data object. access policies are not affected.

check_first
: (logical) Optional. Whether to check the PIDs passed in as arguments exist on the MN before continuing. Checks that objects exist and are of the right format type. This speeds up the function, especially when data_pids has many elements.

format_id
: (character) Optional. When omitted, the updated object will have the same formatId as metadata_pid. If set, will attempt to use the value instead.

Details

This function can be used for a variety of tasks:

- Publish an existing package with a DOI
- Update a package with new data objects
- Update a package with new metadata

The metadata_pid and resource_map_pid provide the identifier of an EML metadata document and associated resource map, and the data_pids vector provides a list of PIDs of data objects in the package. Update the metadata file and resource map by generating a new identifier (a DOI if use_doi = TRUE) and updating the Member Node with a public version of the object. If metadata_file is not missing, it should be an edited version of the metadata to be used to update the original. If parent_resmap_pid is not missing, it indicates the PID of a parent package that should be updated as well, using the parent_metadata_pid, parent_data_pids, and parent_child_pids as members of the updated package. In all cases, the objects are made publicly readable.

Value

(character) Named character vector of PIDs in the data package, including PIDs for the metadata, resource map, and data objects.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
```

```
rm_pid <- "resource_map_urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
meta_pid <- "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
data_pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")

meta_path <- "/home/Documents/myMetadata.xml"

publish_update(mn, meta_pid, rm_pid, data_pids, meta_path, public = TRUE)

## End(Not run)
```

`read_ontology` *Get an owl file from github*

Description

Get an owl file from github

Usage

```
read_ontology(ontology_name)
```

Arguments

`ontology_name`	the name of the onotology to read; one of mosaic or ecso

Value

list

Examples

```
read_ontology("mosaic")
read_ontology("ecso")
```

`read_zip_shapefile` *Read a shapefile from a pid*

Description

Read a shapefile 'sf' from a pid that points to the zipped directory of the shapefile and associated files on a given member node.

Usage

```
read_zip_shapefile(mn, pid)
```

Arguments

mn	(MNode) A DataOne Member Node
pid	(character) An object identifier

Value

shapefile (sf) The shapefile as an sf object

Author(s)

Jeanette Clark jclark@nceas.ucsb.edu

Examples

```
## Not run:
cn <- dataone::CNode('PROD')
adc <- dataone::getMNode(cn,'urn:node:ARCTIC')
pid <- "urn:uuid:294a365f-c0d1-4cc3-a508-2e16260aa70c"

shapefile <- read_zip_shapefile(adc, pid)

## End(Not run)
```

---

recover_failed_submission

Recovers failed submissions

Description

Recovers failed submissions and writes the new, valid EML to a given path

Usage

```
recover_failed_submission(node, pid, path)
```

Arguments

node	(MNode) The Member Node to publish the object to.
pid	The PID of the EML metadata document to be recovered.
path	path to write XML.

Value

recovers and write the valid EML to the indicated path

Author(s)

Rachel Sun rachelsun@ucsb.edu

Examples

```
## Not run:
# Set environment
cn <- dataone::CNode("STAGING2")
mn <- dataone::getMNode(cn,"urn:node:mnTestKNB")
pid <- "urn:uuid:b1a234f0-eed5-4f58-b8d5-6334ce07c010"
path <- tempfile("file", fileext = ".xml")
recover_failed_submission(mn, pid, path)
eml <- EML::read_eml(path)

## End(Not run)
```

---

reformat_file_name *Helper for publish_object. Reformat the fileName in system metadata.*

Description

Reformat the fileName field in an object's system metadata to follow Arctic Data Center system metdata naming conventions. Publish_object calls this function to rename the fileName field in system metadata.

Usage

```
reformat_file_name(path, sysmeta)
```

Arguments

path	(character) full file path
sysmeta	(S4) A system metadata object

---

remove_access *Remove a subject from an object's access policy*

Description

Remove the given subjects from the access policy for the given objects on the given Member Node. For each type of permission, this function checks if the permission is already set and only updates the System Metadata when a change is needed.

Usage

```
remove_access(
  mn,
  pids,
  subjects,
  permissions = c("read", "write", "changePermission")
)
```

Arguments

mn	(MNode) The Member Node.
pids	(character) The PIDs of the objects to set permissions for.
subjects	(character) The identifiers of the subjects to set permissions for, typically an ORCID or DN.
permissions	(character) Optional. The permissions to set. Defaults to read, write, and changePermission.

Value

(logical) Whether an update was needed.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
   "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
remove_access(mn, pids, subjects = "http://orcid.org/0000-000X-XXXX-XXXX",
   permissions = c("read", "write", "changePermission"))

## End(Not run)
```

---

remove_public_read *Remove public read access for an object*

Description

Remove public read access for an object.

Usage

```
remove_public_read(mn, pids)
```

Arguments

mn	(MNode) The Member Node.
pids	(character) The PIDs of the objects to remove public read access for.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
remove_public_read(mn, pids)

## End(Not run)
```

`reorder_pids` *Reorder a named list of objects according to the order in the metadata*

Description

This function takes a named list of data objects, such as what is returned from `get_package`, and reorders them according to the order they are given in the EML document.

Usage

```
reorder_pids(pid_list, doc)
```

Arguments

`pid_list`	(list) A named list of data pids
`doc`	(list) an `emld` document

Value

ordered_pids (list) A list of reordered pids

Examples

```
## Not run:
cn <- dataone::CNode('PROD')
adc <- dataone::getMNode(cn,'urn:node:ARCTIC')
ids <- get_package(adc, 'resource_map_doi:10.18739/A2S17SS1M', file_names = TRUE)
doc <- EML::read_eml(dataone::getObject(adc, ids$metadata))

# return all entity types
ordered_pids <- reorder_pids(ids$data, doc)

## End(Not run)
```

`set_access` *Set the access policy for an object*

Description

Set the access policy for the given subjects for the given objects on the given Member Node. For each type of permission, this function checks if the permission is already set and only updates the System Metadata when a change is needed.

Usage

```
set_access(
  mn,
  pids,
  subjects,
  permissions = c("read", "write", "changePermission")
)
```

Arguments

`mn`	(MNode) The Member Node.
`pids`	(character) The PIDs of the objects to set permissions for.
`subjects`	(character) The identifiers of the subjects to set permissions for, typically an ORCID or DN.
`permissions`	(character) Optional. The permissions to set. Defaults to read, write, and changePermission.

Value

(logical) Whether an update was needed.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
   "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
set_access(mn, pids, subjects = "http://orcid.org/0000-000X-XXXX-XXXX",
   permissions = c("read", "write", "changePermission"))

## End(Not run)
```

set_file_name *Set the file name for an object*

Description

Set the file name for an object.

Usage

```
set_file_name(mn, pid, name)
```

Arguments

mn	(MNode) The Member Node.
pid	(character) The PID of the object to set the file name on.
name	(character) The file name.

Value

(logical) Whether the update succeeded.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn, "urn:node:mnTestKNB")

pid <- "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
set_file_name(mn, pid, "myfile.csv")

## End(Not run)
```

set_public_read *Set public read access for an object*

Description

Set public read access for an object.

Usage

```
set_public_read(mn, pids)
```

Arguments

mn	(MNode) The Member Node.
pids	(character) The PIDs of the objects to set public read access for.

Value

(logical) Whether an update was needed.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
   "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
set_public_read(mn, pids)

## End(Not run)
```

`set_public_read_all_versions` *Set public READ access on all versions of PIDs in data package.*

Description

Set public READ access on all versions of PIDs in data package.

Usage

```
set_public_read_all_versions(mn, resource_map_pid)
```

Arguments

`mn`	(MNode) The Member Node to query.
`resource_map_pid`	(character) The resource map identifier (PID).

Examples

```
## Not run:
cn_staging <- CNode('STAGING')
adc_test <- getMNode(cn_staging,'urn:node:mnTestARCTIC')
# Create a dummy package then create another version with 'publish_update()'
pkg <- create_dummy_package(adc_test)
remove_public_read(mn, unlist(pkg))
pkg_v2 <- publish_update(adc_test, pkg$metadata, pkg$resource_map, pkg$data, public = FALSE)
# Set public read on all versions
set_public_read_all_versions(adc_test, pkg$resource_map)

## End(Not run)
```

`set_rights_and_access` *Set rights holder with access policy for an object*

Description

Set the given subject as the rights holder and with given permissions for the given objects. This function only updates the existing System Metadata when a change is needed.

Usage

```
set_rights_and_access(
  mn,
  pids,
  subject,
  permissions = c("read", "write", "changePermission")
)
```

Arguments

`mn`	(MNode) The Member Node.
`pids`	(character) The PIDs of the objects to set the rights holder and access policy for.
`subject`	(character) The identifier of the new rights holder, typically an ORCID or DN.
`permissions`	(character) Optional. The permissions to set. Defaults to read, write, and changePermission.

Value

(logical) Whether an update was needed.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
    "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
set_rights_and_access(mn, pids, "http://orcid.org/0000-000X-XXXX-XXXX",
    permissions = c("read", "write", "changePermission"))

## End(Not run)
```

set_rights_holder *Set the rights holder for an object*

Description

Set the rights holder to the given subject for the given objects on the given Member Node. This function checks if the rights holder is already set and only updates the System Metadata when a change is needed.

Usage

```
set_rights_holder(mn, pids, subject)
```

Arguments

mn	(MNode) The Member Node.
pids	(character) The PIDs of the objects to set the rights holder for.
subject	(character) The identifier of the new rights holder, typically an ORCID or DN.

Value

(logical) Whether an update was needed.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")
set_rights_holder(mn, pids, subjects = "http://orcid.org/0000-000X-XXXX-XXXX")

## End(Not run)
```

show_indexing_status *Show the indexing status of a set of PIDs*

Description

Show the indexing status of a set of PIDs.

Usage

```
show_indexing_status(mn, pids)
```

Arguments

mn	(MNode) The Member Node to query.
pids	(character/list) One or more PIDs.

Value

NULL

Examples

```
## Not run:
# Create a package then check its indexing status
library(dataone)
mn <- MNode(...)
pkg <- create_dummy_package(mn)
show_indexing_status(mn, pkg)

## End(Not run)
```

---

sysmeta_to_eml_physical

Create an EML physical object from system metadata

Description

This function creates an EML physical object based on what's in the System Metadata of an object. Note that it sets an Online Distribution URL of the DataONE v2 resolve service for the PID.

Usage

```
sysmeta_to_eml_physical(sysmeta)
```

Arguments

sysmeta	(SystemMetadata) One or more System Metadata objects.

Value

(list) A list of physical objects.

Examples

```
## Not run:
# Generate EML physical object from a system metadata object
sm <- getSystemMetadata(mn, pid)
sysmeta_to_eml_physical(sm)

## End(Not run)
```

title_to_file_name *Formats the eml file name based on the dataset title*

Description

Formats the eml file name based on the dataset title

Usage

```
title_to_file_name(title)
```

Arguments

title	(character) title of the dataset

Value

(character) file path with underscores and extension (.xml)

Examples

```
title_to_file_name("Example title here")
```

update_object *Update an object with a new file*

Description

This is a convenience wrapper around `dataone::updateObject()` which copies in fields from the old object's System Metadata such as the rightsHolder and accessPolicy and updates only what needs to be changed.

Usage

```
update_object(mn, pid, path, format_id = NULL, new_pid = NULL, sid = NULL)
```

Arguments

mn	(MNode) The Member Node to update the object on.
pid	(character) The PID of the object to update.
path	(character) The full path to the file to update with.
format_id	(character) Optional. The format ID to set for the object. When not set, `guess_format_id()` will be used to guess the format ID. Should be a DataONE format ID.
new_pid	(character) Optional. Specify the PID for the new object. Defaults to automatically generating a new, random UUID-style PID.
sid	(character) Optional. Specify a Series ID (SID) to use for the new object.

Value

(character) The PID of the updated object.

Examples

```
## Not run:
cn <- CNode("STAGING2")
mn <- getMNode(cn,"urn:node:mnTestKNB")
pid <- "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
my_path <- "/home/Documents/myfile.csv"
new_pid <- update_object(mn, pid, my_path, format_id = "text/csv")

## End(Not run)
```

---

update_resource_map *Update an existing resource map object on a Member Node*

Description

This function first generates a new resource map RDF/XML document locally and then uses the `dataone::updateObject()` function to update an object on the specified MN.

Usage

```
update_resource_map(
  mn,
  resource_map_pid,
  metadata_pid,
  data_pids = NULL,
  child_pids = NULL,
  other_statements = NULL,
  identifier = NULL,
  public = TRUE,
  check_first = TRUE
)
```

Arguments

mn	(MNode) The Member Node.
resource_map_pid	(character) The PID of the resource map to be updated.
metadata_pid	(character) The PID of the metadata object to go in the package.
data_pids	(character) The PID(s) of the data objects to go in the package.
child_pids	(character) The resource map PIDs of the packages to be nested under the package.

other_statements
: (data.frame) Extra statements to add to the resource map.

identifier
: (character) Manually specify the identifier for the new metadata object.

public
: (logical) Whether or not to make the new resource map public read.

check_first
: (logical) Optional. Whether to check the PIDs passed in as arguments exist on the MN before continuing. This speeds up the function, especially when data_pids has many elements.

Details

If you only want to generate resource map RDF/XML, see generate_resource_map().

This function also can be used to add a new child packages to a parent package. For example, if you have:

Parent A B

and want to add C as a sibling package to A and B, e.g.:

Parent A B C

then you could use this function.

Note: This function currently replaces the rightsHolder on the resource map temporarily to allow updating but sets it back to the rightsHolder that was in place before the update.

Value

(character) The PID of the updated resource map.

Examples

```
## Not run:
cn <- CNode('STAGING2')
mn <- getMNode(cn,"urn:node:mnTestKNB")

rm_pid <- "resource_map_urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
meta_pid <- "urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe"
data_pids <- c("urn:uuid:3e5307c4-0bf3-4fd3-939c-112d4d11e8a1",
"urn:uuid:23c7cae4-0fc8-4241-96bb-aa8ed94d71fe")

rm_new <- update_resource_map(mn, rm_pid, meta_pid, data_pids)

## End(Not run)
```

which_in_eml *Search through EMLs*

Description

[Deprecated]

Usage

```
which_in_eml(doc, element, test)
```

Arguments

doc	(list) An EML object.
element	(character) Element to evaluate.
test	(function/character) A function to evaluate (see examples). If test is a character, will evaluate if element == test (see example 1).

Details

please use eml_get_simple() and which() together instead

This function returns indices within an EML list that contain an instance where test == TRUE. See examples for more information.

Author(s)

Mitchell Maier mitchell.maier@gmail.com

Examples

```
## Not run:
# Question: Which creators have a surName "Smith"?
n <- which_in_eml(eml$dataset$creator, "surName", "Smith")
# Answer: eml$dataset$creator[n]

# Question: Which dataTables have an entityName that begins with "2016"
n <- which_in_eml(eml$dataset$dataTable, "entityName", function(x) {grepl("^2016", x)})
# Answer: eml$dataset$dataTable[n]

# Question: Which attributes in dataTable[[1]] have a numberType "natural"?
n <- which_in_eml(eml$dataset$dataTable[[1]]$attributeList$attribute, "numberType", "natural")
# Answer: eml$dataset$dataTable[[1]]$attributeList$attribute[n]

#' # Question: Which dataTables have at least one attribute with a numberType "natural"?
n <- which_in_eml(eml$dataset$dataTable, "numberType", function(x) {"natural" %in% x})
# Answer: eml$dataset$dataTable[n]

## End(Not run)
```

Index