Package ‘dataone’

October 13, 2022

Version 2.2.2
Date 2022-06-08
Title R Interface to the DataONE REST API
Description Provides read and write access to data and metadata from the DataONE network <https://www.dataone.org> of data repositories. Each DataONE repository implements a consistent repository application programming interface. Users call methods in R to access these remote repository functions, such as methods to query the metadata catalog, get access to metadata for particular data packages, and read the data objects from the data repository. Users can also insert and update data objects on repositories that support these methods.

Depends R (>= 3.1.1)
Suggests knitr, rmarkdown, testthat, digest, openssl (>= 0.9.3), xml2
Imports XML (>= 3.95-0.1), httr, methods, stringi, stringr, datapack (>= 1.4.0), plyr, parsedate, uuid, base64enc, jsonlite
License Apache License 2.0
URL https://github.com/DataONEorg/rdataone
BugReports https://github.com/DataONEorg/rdataone/issues
Collate 'AbstractTableDescriber.R' 'auth_request.R' 'D1Node.R'
 'AuthenticationManager.R' 'CNode.R' 'CertificateManager.R'
 'D1Object.R' 'MNode.R' 'D1Client.R' 'EMLParser.R'
 'dataone-defunct.R' 'dataone-deprecated.R' 'dataone-package.R'

Encoding UTF-8
VignetteBuilder knitr
RoxygenNote 7.1.2
NeedsCompilation no

Author Matthew B. Jones [aut, cre] (<https://orcid.org/0000-0003-0077-4738>),
Peter Slaughter [aut] (<https://orcid.org/0000-0002-2192-403X>),
Rob Nahf [aut],
Carl Boettiger [aut] (<https://orcid.org/0000-0002-1642-628X>),
Chris Jones [aut] (<https://orcid.org/0000-0002-8121-2341>),
Bryce Mecum [aut] (<https://orcid.org/0000-0002-0381-3766>),
Jeanette Clark [aut] (<https://orcid.org/0000-0003-4703-1974>),
Jordan Read [ctb] (<https://orcid.org/0000-0002-3888-6631>),
Lauren Walker [aut] (<https://orcid.org/0000-0003-2192-431X>),
Edmund Hart [ctb] (<https://orcid.org/0000-0001-7367-7969>),
Scott Chamberlain [ctb] (<https://orcid.org/0000-0003-1444-9135>),
Regents of the University of California [cph]

Maintainer Matthew B. Jones <jones@nceas.ucsb.edu>
Repository CRAN
Date/Publication 2022-06-10 19:30:02 UTC

R topics documented:

AbstractTableDescriber-class ........................................ 4
addData,DataPackage,D1Object-method ................................ 5
archive ................................................................. 6
asDataFrame ............................................................. 7
AuthenticationManager ..................................................... 8
AuthenticationManager-class ............................................. 9
auth_delete .............................................................. 10
auth_get ................................................................. 11
auth_head ................................................................. 11
auth_post ................................................................. 12
auth_put ................................................................. 13
auth_put_post_delete ................................................... 13
canRead,D1Object-method ............................................. 14
CertificateManager ....................................................... 14
CertificateManager-class .............................................. 15
CNode ...................................................................... 17
CNode-class ............................................................. 18
convert.csv .............................................................. 19
createD1Object ........................................................ 20
createDataPackage ..................................................... 21
createObject ............................................................. 22
D1Client ................................................................. 23
D1Client-class .......................................................... 24
d1IdentifierSearch ..................................................... 25
D1Node ................................................................. 26
D1Node-class ............................................................ 27
D1Object ................................................................. 28
D1Object-class .......................................................... 28
d1SolrQuery .............................................................. 29
d1_errors ................................................................. 30
data.characterEncoding ............................................... 30
data.formatFamily ...................................................... 31
data.tableAttributeNames ........................................... 31
data.tableAttributeOrientation ..................................... 32

data.tableAttributeStorageTypes ........................................... 33

data.tableAttributeTypes .................................................. 33

data.tableFieldDelimiter .................................................. 34

data.tableMissingValueCodes .............................................. 35

data.tableQuoteCharacter .................................................. 35

data.tableSkipLinesHeader ............................................... 36

dataone .................................................................. 37

describeObject ................................................................. 37

documented.d1Identifiers .................................................. 38

documented.entityNames ................................................... 39

documented.sizes ............................................................. 40

downloadCert ................................................................. 40

downloadObject .................................................................. 41

echoCredentials .............................................................. 42

EMLParser ................................................................. 43

EMLParser-class ............................................................... 43

encodeSolr ................................................................. 44

encodeUrlPath ............................................................... 44

encodeUrlQuery ............................................................... 45

evaluateAuth ................................................................. 46

generateIdentifier .......................................................... 47

getAuthExpires ............................................................... 48

getAuthMethod ............................................................... 48

getAuthSubject ............................................................... 49

getCapabilities ............................................................. 50

getCert ................................................................. 51

getCertExpires .............................................................. 51

getCertInfo ................................................................. 52

getCertLocation ............................................................. 52

getChecksum ................................................................. 53

getCN ................................................................. 54

getD1Object ................................................................. 55

getData,D1Object-method ................................................. 56

dataObject ................................................................. 56

dataPackage ................................................................. 58

getEndpoint ................................................................. 59

getErrorDescription ........................................................ 60

getFormat ................................................................. 61

getFormatId,D1Object-method ............................................. 62

getIdentifer,D1Object-method ............................................. 62

getMetadataMember ........................................................ 63

getMN ................................................................. 63

getMNode ................................................................. 64

getMNodeId ................................................................. 65

getObject ................................................................. 66

getPackage ................................................................. 67

dataQueryEngineDescription ............................................... 68

getSystemMetadata ........................................................ 69
AbstractTableDescriber-class

Base Class for Specific Metadata Parsers

Description

Classes that inherit from this class provide the format-specific ways to provide read.csv with parsing instructions.
addData, DataPackage, D1Object-method

Details

This class defines the generic methods metadata parser classes need to implement to allow proper parsing of tabular data objects. Subclasses should: 1. provide method implementations for all generics 2. register the class to the tableDescriber.registry for the formats they claim to parse. 3. provide a "constructor" method that accepts a D1Object as the first argument - the D1Object will be the metadata object to be parsed

For example, the EMLParser registers itself as a handler for eml v2.0.0 - v2.1.1 with the following.

```r
if (!exists("tableDescriber.registry")) tableDescriber.registry <- list()
tableDescriber.registry[[ "eml://ecoinformatics.org/eml-2.0.0"]]<- "EMLParser"
tableDescriber.registry[[ "eml://ecoinformatics.org/eml-2.1.0"]]<- "EMLParser"
tableDescriber.registry[[ "eml://ecoinformatics.org/eml-2.1.1"]]<- "EMLParser"
```

Note that the key in the list is the DataONE formatIdentifier that can be found at "https://cn.dataone.org/cn/v2/formats".

Subclass implementers should conform their methods to the behavior defined in the generic.

Author(s)

rnahf

---

addData(DataPackage, D1Object-method)

Add a D1Object containing a data object to a DataPackage

Description

The D1Object do is added to the data package x.

Usage

```r
## S4 method for signature 'DataPackage,D1Object'
addData(x, do, mo = as.character(NA))
```

Arguments

- **x**: The "DataPackage" to which the data object should be added.
- **do**: A D1Object to add to the DataPackage
- **mo**: A D1Object (containing metadata describing "do") to associate with the data object.

Details

If the optional mo parameter is specified, then it is assumed that this DataObject is a metadata object that describes the data object that is being added. The DataObject specified in the mo parameter will also be added to the DataPackage, if it has not already been added. Then the addData function will add a relationship to the resource map that indicates that the metadata object describes the science object, using CiTO, the Citation Typing Ontology, documents and isDocumentedBy relationships.
Examples

    ## Not run:
    library(dataone)
    library(datapack)
    library(uuid)
    dp <- new("DataPackage")
    d1c <- D1Client("STAGING", "urn:node:mmStageUCSB2")
    # Create metadata object that describes science data
    newId <- sprintf("urn:uuid:%s", UUIDgenerate())
    csvfile <- system.file("extdata/sample.csv", package="dataone")
    sciObj <- new("DataObject", id=newId, format="text/csv", filename=csvfile)
    dp <- addData(dp, do = sciObj)

    ## End(Not run)

archive

Archive an object on a Member Node or Coordinating Node, which hides it from casual searches.

Description

This method provides the ability to archive a data or metadata object on the Member Node provided in the 'mNode' parameter. Archiving removes the object from DataONE search functions, thereby making it more difficult to find without completely removing the object. Archive is intended for objects that should not be used by current researchers, but for which there is a desire to maintain a historical record, such as when journal articles might cite the object. Users can still obtain the contents of archived objects if they have the identifier, but will not discover it through searches.

Usage

    archive(x, ...)

    ## S4 method for signature 'D1Node'
    archive(x, pid)

Arguments

    x       The MNode or CNode instance on which the object will be created
    ...     (Not yet used)
    pid     The identifier of the object to be created

Details

This operation requires an X.509 certificate to be present in the default location of the file system. This certificate provides authentication credentials from CILogon https://cilogon.org/?skin=DataONE. See CertificateManager for details. For DataONE Version 2.0, an authentication token can also be used for authentication. Also, administrator privilege is required to run archive() on a DataONE Coordinating Node.
Value

The pid that was archived if successful, otherwise NULL

See Also

D1Node class description.

Examples

```r
## Not run:
library(dataone)
library(uuid)
library(digest)
library(datapack)
# First create a new object
cn <- CNode("STAGING")
mn <- getMNode(cn, "urn:node:mmStageUCSB2")
testdf <- data.frame(x=1:10, y=11:20)
csvfile <- paste(tempfile(), ".csv", sep="")
write.csv(testdf, csvfile, row.names=FALSE)
\dontrun{
  newid <- generateIdentifier(mn, "UUID")
}
# Create an identifier manually
newid <- paste("urn:uuid:", UUIDgenerate(), sep="")
format <- "text/csv"
size <- file.info(csvfile)$size
sha256 <- digest(csvfile, algo="sha256", serialize=FALSE, file=TRUE)
sysmeta <- new("SystemMetadata", identifier=newid, formatId=format, size=size, checksum=sha256)
sysmeta <- addAccessRule(sysmeta, "public", "read")
# Create (upload) the object to DataONE (requires authentication)
\dontrun{
  create(mn, newid, csvfile, sysmeta)
  # Now for demonstration purposes, archive the object
  # Archive the object (requires authentication)
  archivedId <- archive(mn, newid)
}
## End(Not run)
```

---

**asDataFrame**

Return the D1Object data as a data.frame.

**Description**

This method uses the provided metadata reference object for instructions on how to parse the data table (which parameters to set) ‘reference’ is the metadata D1Object that gives instruction on how to read the data into the dataFrame
Usage

asDataFrame(x, reference, ...)

## S4 method for signature 'D1Object,D1Object'
asDataFrame(x, reference, ...)

## S4 method for signature 'D1Object,AbstractTableDescriber'
asDataFrame(x, reference, ...)

Arguments

x A D1Object
reference A reference to a D1Object
... (Additional parameters)

AuthenticationManager Create an AuthenticationManager object

Description

Construct an instance of AuthenticationManager to provide mechanisms to load, verify, and display DataONE authentication information.

Usage

AuthenticationManager(...)

## S4 method for signature 'ANY'
AuthenticationManager()

Arguments

... (Not yet used)

Value

the AuthenticationManager object
AuthenticationManager-class

Manage DataONE authentication.

Description

AuthenticationManager provides mechanisms to validate DataONE authentication, when either a DataONE authentication token or X.509 Certificate is used.

Details

Understanding how your identity is managed is important for working with DataONE, especially to avoid unexpected results. For example, depending your authorization status, searches may return only public records, or the full set of public and private records. Object and package retrievals might fail if some or all of the objects being retrieved are private. Creating or updating objects on DataONE nodes and reserving identifiers might fail if your authorization credentials are missing or expired.

DataONE version 1.0 identifies you using CILogon-provided x509 certificates. DataONE has partnered with CILogon to provide a widely-accessible certificate issuing mechanism that allows DataONE users to use existing trusted institutional and public accounts.

DataONE version 2.0 provides an addition authentication mechanism known as authentication tokens. For information about tokens and instructions for generating a token for use with the dataone R package, view the overview document by entering the command: `vignette("dataone-overview")`.

DataONE authentication tokens can be obtained by signing in to your DataONE account at https://search.dataone.org. CILogon recognizes many identity providers, including many universities as well as Google, so most times users new to DataONE can get certificates using one of their existing accounts. For more information about the CILogon service, see https://cilogon.org/?skin=DataONE.

Slots

- `obscured` Value of type "character" Is authentication disabled (obscured)?

Methods

- `AuthenticationManager`: Create an AuthenticationManager object.
- `isAuthValid`: Verify authentication for a member node.
- `getToken`: Get the value of the DataONE Authentication Token, if one exists.
- `getCert`: Get the DataONE X.509 Certificate location.
- `getAuthMethod`: Get the current valid authentication mechanism.
- `getAuthSubject`: Get the authentication subject.
- `getAuthExpires`: Get the expiration date of the current authentication method.
- `isAuthExpired`: Check if the currently valid authentication method has reached the expiration time.
- `obscureAuth`: Temporarily disable DataONE authentication.
- **restoreAuth**: Restore authentication (after being disabled with `obscureAuth`).
- **showAuth**: Display all authentication information.
- **getTokenInfo**: Display all authentication token information.
- **getCertInfo**: Display all X.509 certificate information.

**See Also**

`dataone` package description.

---

**auth_delete**

DELETE a resource with authenticated credentials.

**Description**

DELETE data at a URL using an HTTP DELETE request using authentication credentials provided in a client certificate. Authenticated access depends on the suggested openssl package. If the openssl package is not installed, then the request fails.

**Usage**

```r
auth_delete(url, encode = "multipart", body = as.list(NA), node)
```

**Arguments**

- **url**
  - The URL to be accessed via authenticated DELETE
- **encode**
  - the type of encoding to use for the DELETE body, defaults to `"multipart`'
- **body**
  - a list of data to be included in the body of the DELETE request
- **node**
  - The D1Node object that the request will be made to.

**Value**

the HTTP response from the request
auth_get

**Description**

Retrieve the data at a URL using an HTTP GET request using authentication credentials provided in a client certificate. Authenticated access depends on the suggested openssl package. If the openssl package is not installed, then the request falls back to an unauthenticated request, which may fail due to insufficient permissions. Configuration options for httr/Rcurl can be passed using the normal config() mechanisms to generate a config option. Use httr_options() to see a complete list of available options.

**Usage**

```r
auth_get(url, nconfig = config(), node, path = NULL)
```

**Arguments**

- `url`: The URL to be accessed via authenticated GET.
- `nconfig`: HTTP configuration options as used by curl, defaults to empty list
- `node`: The D1Node object that the request will be made to.
- `path`: Path to a file to write object to

**Value**

the response object from the method

---

auth_head

**Description**

Retrieve http header information for a URL using an HTTP HEAD request using authentication credentials provided in a client certificate or token. Authenticated access depends on the suggested openssl package. If the openssl package is not installed, then the request falls back to an unauthenticated request, which may fail due to insufficient permissions. Configuration options for httr/Rcurl can be passed using the normal config() mechanisms to generate a config option. Use httr_options() to see a complete list of available options. Note: The HEAD method is identical to GET except that the server MUST NOT return a message-body in the response.

**Usage**

```r
auth_head(url, nconfig = config(), node)
```
auth_post

Arguments

url The URL to be accessed via authenticated HEAD.
nconfig HTTP configuration options as used by curl, defaults to empty list
node The D1Node object that the request will be made to.

Value

the response object from the method

auth_post

**POST a resource with authenticated credentials.**

Description

POST data to a URL using an HTTP POST request using authentication credentials provided in a client certificate. Authenticated access depends on the suggested openssl package. If the openssl package is not installed, then the request fails.

Usage

auth_post(url, encode = "multipart", body = NULL, node)

Arguments

url The URL to be accessed via authenticated POST
encode the type of encoding to use for the POST body, defaults to `multipart`
body a list of data to be included in the body of the POST request
node The D1Node object that the request will be made to.

Value

the HTTP response from the request
auth_put

PUT a resource with authenticated credentials.

**Description**

PUT data to a URL using an HTTP PUT request using authentication credentials provided in a client certificate. Authenticated access depends on the suggested openssl package. If the openssl package is not installed, then the request fails.

**Usage**

```
auth_put(url, encode = "multipart", body = NULL, node)
```

**Arguments**

- **url**: The URL to be accessed via authenticated PUT
- **encode**: the type of encoding to use for the PUT body, defaults to 'multipart'
- **body**: a list of data to be included in the body of the PUT request
- **node**: The D1Node object that the request will be made to.

**Value**

the HTTP response from the request

---

auth_put_post_delete

POST, PUT, or DELETE a resource with authenticated credentials.

**Description**

POST, PUT, or DELETE data to a URL using an HTTP request using authentication credentials provided in a client authentication, either via authentication token or certificate. If the user does not have a valid token or certificate, request fails.

**Usage**

```
auth_put_post_delete(method, url, encode = "multipart", body = NULL, node)
```

**Arguments**

- **method**: a string indicating which HTTP method to use (post, put, or delete)
- **url**: The URL to be accessed via authenticated PUT
- **encode**: the type of encoding to use for the PUT body, defaults to 'multipart'
- **body**: a list of data to be included in the body of the PUT request
- **node**: The D1Node object that the request will be made to.
CertificateManager

Value
the response object from the method

canRead,D1Object-method
Test whether the provided subject can read an object.

Description
Using the AccessPolicy, tests whether the subject has read permission for the object. This method is meant work prior to submission to a repository, and will show the permissions that would be enforced by the repository on submission. Currently it only uses the AccessPolicy to determine who can read (and not the rightsHolder field, which always can read an object). If an object has been granted read access by the special "public" subject, then all subjects have read access.

Usage
## S4 method for signature 'D1Object'
canRead(x, subject)

Arguments
x D1Object
subject : the subject name of the person/system to check for read permissions

Details
The subject name used in both the AccessPolicy and in the 'subject' argument to this method is a string value, but is generally formatted as an X.509 name formatted according to RFC 2253.

Value
logical TRUE if the subject has read permission, or FALSE otherwise

CertificateManager Create a CertificateManager object

Description
Construct an instance of CertificateManager to provide mechanisms to obtain, load, verify, and display X509 certificates. If the 'location' field is provided, then that location is interpreted as the fully qualified path to a certificate on the local filesystem, and the default locations will not be searched. If 'location' is missing, then the default Globus Grid Security Infrastructure (GSI) location is searched, which is '/tmp/x509up_u${UID}' on Unix or '${tmpdir}/x509up_u${UID}' on Windows or '${tmpdir}/x509up_u${user.name}' if '${UID}' is not defined.
CertificateManager-class

CertificateManager-class

CertificateManager provides mechanisms to obtain, load, verify, and display X509 certificates.

Description

CertificateManager provides management functions for X.509 certificates that are used to authenticate connections to DataONE nodes over SSL. The X.509 certificates are issued by a recognized Certificate Authority, typically CILogon, and include fields that provide information about the authenticated party, including the distinguished name of the subject, the dates of validity of the certificate, and other information needed for authorization decisions. Certificate validity is determined by examining the validity of the certificate signatures for each certificate in a chain leading to a trusted root certificate. Within DataONE, the current trusted root certificate authorities are CILogon and DataONE itself.

Details

Understanding how your identity is managed is important for working with DataONE, especially to avoid unexpected results. For example, depending your authorization status, searches may or may return only public records, or the full set of public and private records. Object and package retrievals might fail if some or all of the objects being retrieved are private. Creating or updating objects on DataONE nodes and reserving identifiers reservations might fail if your authorization certificate is missing or expired.

DataONE identifies you using CILogon-provided x509 certificates. DataONE has partnered with CILogon to provide a widely-accessible certificate issuing mechanism that allows DataONE users to use existing trusted institutional and public accounts.

CILogon recognizes many identity providers, including many universities as well as Google, so most times users new to DataONE can get certificates using one of their existing accounts. For more information about the CILogon service, see “https://cilogon.org/?skin=DataONE”.

Usage

CertificateManager(...)  

## S4 method for signature 'ANY'
CertificateManager()

Arguments

... (Not yet used)

Value

the CertificateManager object
X509 Certificates differ from typical username-password login schemes in that certificates can be used by more than one application, which is very useful when using more than one DataONE-enabled application. The certificates CILogon issues for DataONE are so-called "short-lived" certificates that currently expire 18 hours from the time of issuing. Typically you will want to download a fresh certificate the first time you interact with DataONE each day.

Slots

- **location**: value of type "character", containing a path to a custom certificate location
- **obscuredpath**: value of type "character", containing the path used to temporarily obscure a certificate

Methods

- **CertificateManager**: Create a CertificateManager object.
- **getCertLocation**: Get the file path on disk of the client certificate file.
- **showClientSubject**: Get DataONE Identity as Stored in the CILogon Certificate.
- **isCertExpired**: Determine if an X.509 certificate has expired.
- **getCertExpires**: Show the date and time when an X.509 certificate expires.
- **downloadCert**: Open the CILogon Certificate download page in the default browser.
- **obscureCert**: Obscure the CILogon Client Certificate.
- **restoreCert**: Restore the CILogon client certificate by renaming it to its original location

Author(s)

Matthew Jones, Rob Nahf

See Also

dataone package description.

Examples

```r
## Not run:
cm <- suppressWarnings(CertificateManager())
cert <- getCertLocation(cm)
subject <- showClientSubject(cm)
expires <- getCertExpires(cm)
isExpired <- isCertExpired(cm)
cm <- obscureCert(cm)
cm <- restoreCert(cm)

## End(Not run)
```
Create a CNode object.

Usage

CNode(x, ...)

## S4 method for signature 'ANY'
CNode()

## S4 method for signature 'character'
CNode(x)

Arguments

x

The label for the DataONE environment to be using ('PROD', 'STAGING', 'STAGING2', 'SANDBOX', 'SANDBOX2', 'DEV', 'DEV2')

... (not yet used)

Details

For an explanation of DataONE Coordinating Nodes, see the section "DataONE Environments" in the overview vignette by entering the R command: vignette("dataone-overview").

Value

the CNode object representing the DataONE environment

See Also

CNode class description.

Examples

## Not run:
cn <- CNode("PROD")

## End(Not run)
CNode-class

Provides R API to DataONE Coordinating Node services.

Description

The CNode class provides methods that interact with a DataONE Coordinating Node.

Slots

description A character vector containing URL service endpoint for the Coordinating Node

descriptions A data.frame containing the supported service tiers for a CN

deserveUrls A data.frame contains URL endpoints for certain services

Methods

- **CNode**: Construct a CNode object.
- **listFormats**: List all object formats registered in DataONE.
- **getFormat**: Get information for a single DataONE object format
- **getChecksum**: Get the checksum for the data object associated with the specified pid.
- **listNodes**: Get the list of nodes associated with a CN.
- **reserveIdentifier**: Reserve a identifier that is unique in the DataONE network.
- **hasReservation**: Checks to determine if the supplied subject is the owner of the reservation of id.
- **setObsoletedBy**: Set a pid as being obsoleted by another pid
- **getObject**: Get the bytes associated with an object on this Coordinating Node.
- **getSystemMetadata**: Get the bytes associated with an object on this Coordinating Node.
- **describeObject**: Get a list of coordinating nodes holding a given pid.
- **resolve**: Get a list of coordinating nodes holding a given pid.
- **getMNode**: Get a reference to a node based on its identifier.
- **echoCredentials**: Echo the credentials used to make the call.
- **isAuthorized**: Check if an action is authorized for the specified identifier.

See Also

dataone package description.
**convert.csv**

*Convert a DataFrame to Standard CSV.*

---

### Description

Convert a DataFrame to Standard CSV.

### Usage

```r
convert.csv(x, ...)  
```

```r  
## S4 method for signature 'D1Client'  
convert.csv(x, df, ...)  
```

### Arguments

- **x**: A D1Client object
- **...**: additional params passed to write.csv
- **df**: the dataFrame

### Value

the dataframe serialized as a .csv

### See Also

*D1Client* class description.

### Examples

```r  
## Not run:  
d1c <- D1Client("STAGING", "urn:node:mmStageUCSB2")  
testdf <- data.frame(x=1:10,y=11:20)  
sdf <- convert.csv(d1c, testdf)  
## End(Not run)  
```
createD1Object  

Create the Object in the DataONE System

Description

Create the Object in the DataONE System

Usage

createD1Object(x, d1Object, ...)

## S4 method for signature 'D1Client,D1Object'
createD1Object(x, d1Object)

Args

x : D1Client
d1Object : A D1Object instance to upload to DataONE
... (not yet used)

Value

TRUE if the object was successfully uploaded, FALSE if not.

Examples

## Not run:
library(dataone)
library(uuid)
d1c <- D1Client("STAGING", "urn:node:mnStageUCSB2")
data <- readLines(system.file("extdata/strix-pacific-northwest.xml", package="dataone"))
dataRaw <- charToRaw(paste(data, collapse="\n"))
newid <- sprintf("urn:node:%s", UUIDgenerate())
d1o <- new("D1Object", id=newid, data=dataRaw, format="text/plain")
d1o <- setPublicAccess(d1o)
# Upload the object to DataONE (requires authentication)
uploaded <- createD1Object(d1c, d1o)

## End(Not run)
createDataPackage  

Create a DataPackage on a DataONE Member Node

Description

Upload all members of a DataPackage to DataONE.

Usage

createDataPackage(x, dataPackage, ...)

## S4 method for signature 'D1Client,DataPackage'
createDataPackage(x, dataPackage, ...)

Arguments

x  A D1Client instance.

dataPackage  The DataPackage instance to be submitted to DataONE for creation.

...  Additional arguments

Value

The identifier of the uploaded package.

See Also

D1Client class description.

Examples

## Not run:
library(dataone)
testdf <- data.frame(x=1:10, y=11:20)
csvfile <- tempfile(pattern = "file", tmpdir = tempdir(), fileext = ".csv")
write.csv(testdf, csvfile, row.names=FALSE)
dlnc <- D1Client("STAGING", "urn:node:mmStageUCSB2")
dp <- new("DataPackage")
emlFile <- system.file("extdata/strix-pacific-northwest.xml", package="dataone")
emlChar <- readLines(emlFile)
emlRaw <- charToRaw(paste(emlChar, collapse="\n"))
emlId <- sprintf("urn:uuid:%s", UUIDgenerate())
metadataObj <- new("D1Object", id=emlId, format="eml://ecoinformatics.org/eml-2.1.1", data=emlRaw,
mnNodeId=getMNodeId(dlnc))
sdf <- read.csv(csvfile)
stf <- charToRaw(convert.csv(dlnc, sdf))
scvid <- sprintf("urn:uuid:%s", UUIDgenerate())
sciObj <- new("D1Object", id=scvid, format="text/csv", data=stf, mnNodeId=getMNodeid(dlnc))
dp <- addMember(dp, do=sciObj, mo=metadataObj)
createObject

Create an object on a Member Node.

Description

This method provides the ability to upload a data or metadata object to the Member Node provided in the 'mnode' parameter.

Usage

createObject(x, ...)

## S4 method for signature 'MNode'
createObject(x, pid, file = as.character(NA), sysmeta, dataobj = NULL, ...)

Arguments

- **x**: The MNode instance on which the object will be created
- **...**: (Not yet used.)
- **pid**: The identifier of the object to be created
- **file**: The absolute file location of the object to be uploaded
- **sysmeta**: a SystemMetadata instance describing properties of the object
- **dataobj**: a raw object to use for the upload, instead of the contents of the file argument.

Details

In the version 2.0 library and higher, this operation can utilize an 'dataone_token' option to provide credentials for write operations in DataONE. The authentication token is obtained from DataONE (see your profile on https://search.dataone.org). See the vignette("dataone-overview") for details. Alternatively, the version 1.0 approach of using an X.509 certificate in a default location of the file system can also be used. This certificate provides authentication credentials from CILogon https://cilogon.org/?skin=DataONE. See vignette("dataone-overview") for details.

Value

a character containing the identifier that was created.

See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MNStorage.create
## Examples

```r
## Not run:
# Create an object in the DataONE "STAGING" environment
library(dataone)
library(uuid)
library(digest)
library(datapack)
cn <- CNode("STAGING")
mn <- getMNode(cn, "urn:node:mmStageUCSB2")
# Have Dataone create an identifier for you (requires authentication)
\dontrun{
  newid <- generateIdentifier(mn, "UUID")
}
# Create an identifier manually
newid <- paste("urn:uuid:", UUIDgenerate(), sep="")
testdf <- data.frame(x=1:10, y=11:20)
csvfile <- paste(tempfile(), ".csv", sep="")
write.csv(testdf, csvfile, row.names=FALSE)
format <- "text/csv"
size <- file.info(csvfile)$size
sha256 <- digest(csvfile, algo="sha256", serialize=FALSE, file=TRUE)
sysmeta <- new("SystemMetadata", identifier=newid, formatId=format, size=size, checksum=sha256)
sysmeta <- addAccessRule(sysmeta, "public", "read")
# Upload the data to DataONE (requires authentication)
\dontrun{
  createObject(mn, newid, csvfile, sysmeta)
}

## End(Not run)
```

---

**D1Client**  
The DataONE client class used to download, update and search for data in the DataONE network.

### Description

The DataONE client class used to download, update and search for data in the DataONE network.

### Usage

```r
D1Client(x, y, ...)
```

## S4 method for signature 'ANY,ANY'

```r
D1Client()
```

## S4 method for signature 'character,ANY'

```r
D1Client(x, y, ...)
```

## S4 method for signature 'character,character'

```r
D1Client(x, y, ...)
```
D1Client(x, y)

## S4 method for signature 'CNode,MNode'
D1Client(x, y, ...)

## S4 method for signature 'character,MNode'
D1Client(x, y, ...)

**Arguments**

- **x**
  - The label for the DataONE environment to be using ('PROD', 'STAGING', 'SANDBOX', 'DEV'). This parameter can alternatively be a `CNode` instance, with the 'y' parameter specified as an `MNode` instance.

- **y**
  - The node Id of the application’s 'home' node. Should be already registered to the corresponding 'env'. This parameter can alternatively be an `MNode` instance, with the 'x' parameter specified as a `CNode` instance.

- **...**
  - (not yet used)

**Value**

the D1Client object representing the DataONE environment

**See Also**

D1Client class description.

**Examples**

```r
## Not run:
c1i <- D1Client("PROD", "urn:node:KMB")
cn <- CNode("STAGING")
mn <- getMNode(cn, "urn:node:mnTestKMB")
c1i <- D1Client(cn,mn)
## End(Not run)
```

---

**D1Client-class**

The D1Client class contains methods that perform high level DataONE tasks

**Description**

The methods in the D1Client class call the low level DataONE API to perform involved tasks such as uploading all the packages in a DataPackage (i.e uploadDataPackage)

**Slots**

- **cn** The Coordinating Node associated with the D1Client object
- **mn** The Member Node associated with this D1Client object
Methods

- **D1Client**: Construct a D1Client object.
- **convert.csv**: Convert a DataFrame to Standard CSV.
- **createDataPackage**: Create a DataPackage on a DataONE Member Node.
- **encodeUrlPath**: Encode the Input for a URL Path Segment.
- **encodeUrlQuery**: Encode the Input for a URL Query Segment.
- **getDataObject**: Download a single data object from a DataONE Federation member node.
- **getDataPackage**: Download a collection of data object from the DataONE Federation member node as a DataPackage.
- **getEndpoint**: Return the URL endpoint for the DataONE Coordinating Node.
- **getMetadataMember**: Get the DataObject containing package metadata.
- **getMNodeId**: Get the member node identifier associated with this D1Client object.
- **listMemberNodes**: List DataONE Member Nodes.
- **reserveIdentifier**: Reserve a unique identifier in the DataONE Network.
- **uploadDataObject**: Upload a DataObject to a DataONE member node.
- **uploadDataPackage**: Upload a DataPackage to a DataONE member node.

See Also

`dataone` package description.

d1IdentifierSearch

`d1IdentifierSearch` *Query the DataONE Solr endpoint of the Coordinating Node.*

Description

The DataONE CN Solr query engine is searched using the provided query string.

Usage

```r
d1IdentifierSearch(x, ...)
```

## S4 method for signature 'D1Client'
d1IdentifierSearch(x, solrQuery)

Arguments

- `x` D1Client: representing the DataONE environment being queried
- `...` Additional parameters
- `solrQuery` character: a query string
D1Node

Description

Create a D1Node object.

Usage

D1Node(xml, ...)

## S4 method for signature 'XMLInternalElementNode'
D1Node(xml)

Arguments

xml An XML object that describes the node to be initialized (see listNodes).
...
(not yet used)

Value

the Node object representing the DataONE environment
D1Node-class

A base class for CNode and MNode.

Description

D1Node is a base class for CNode and MNode classes and contains class slots and methods that are common between these two child classes.

Slots

- **identifier** A character string containing a URN that uniquely identifies the node
- **name** A character string containing a plain text name for the node
- **description** A character string describing the node
- **baseURL** A character string of the registered baseURL for the node, which does not include the version string
- **subject** A character string containing the Distinguished Name of this node, used for authentication
- **contactSubject** The Distinguished Name of contact person for this node
- **replicate** A logical flag indicating whether the node accepts replicas
- **type** The node type, either 'mn' or 'cn'
- **state** A character string that indicates whether or not the node is accessible, either 'up' or 'down'
- **services** A data.frame containing the service tiers supported by this node.
- **serviceUrls** A data.frame that contains DataONE service URLs
- **APIversion** A character string indicating version of the DataONE API for this node, e.g. "v2"
- **env** A character string, either 'prod' if this node is in the production environment, otherwise 'test'

Methods

- **D1Node-initialize**<i>{initialize}</i>: Initialize a D1Node
- **D1Node**: Create a MNode object representing a DataONE Member Node repository.
- **archive**: Change the state of an object so that it is hidden from searches.
- **describeObject**: Get header information for a given pid.
- **getChecksum**: Get the checksum for the data object associated with the specified pid.
- **getObject**: Get the bytes associated with an object on a node.
- **getQueryEngineDescription**: Query a node for the list of query engines available on the node.
- **getSystemMetadata**: Get the metadata describing system properties associated with an object on the Node.
- **listObjects**: Retrieve the list of objects that match the search parameters.
- **listQueryEngines**: Query a node for the list of query engines available on the node.
- **ping**: Test if a node is online and accepting DataONE requests.
- **encodeSolr**: Encode the input for Solr Queries.
- **query**: Search DataONE for data and metadata objects.
- **isAuthorized**: Check if an action is authorized for the specified identifier.
D1Object

Create a D1Object instance.

Description
Create a D1Object instance.

Usage
D1Object(...)

Arguments
... (additional arguments)

Value
the D1Object instance

See Also
D1Object class description.

D1Object-class
D1Object (Defunct) is a representation of a DataObject.

Description
D1Object has been defunct in favor of datapack::DataObject, which provides a wrapper for data and associated SystemMetadata.

Slots
dataObject A backing instance of a DataObject, to which all methods and state are proxied

Methods
- **D1Object-initialize**: Initialize a D1Object
- **getData**: Get the data content of a specified D1Object.
- **getIdentifier**: Get the identifier of the D1Object.
- **getFormatId**: Get the formatId of the D1Object
- **setPublicAccess**: Add a Rule to the AccessPolicy to make the object publicly readable.
- **canRead**: Test whether the provided subject can read an object.
- **asDataFrame**: Return the D1Object as a data.frame.
d1SolrQuery

See Also

dataone package description.

---

d1SolrQuery  A method to query the DataONE solr endpoint of the Coordinating Node.

Description

It expects any lucene reserved characters to already be escaped with backslash. If solrQuery is a list, it is expected to have field names as attributes and search values as the values in the list.

Usage

d1SolrQuery(x, solrQuery)

## S4 method for signature 'D1Client,list'
d1SolrQuery(x, solrQuery)

## S4 method for signature 'D1Client,character'
d1SolrQuery(x, solrQuery)

Arguments

x  the D1Client (environment) being queried

solrQuery  list or character: a fully encoded query string

Value

the solr response (XML)

See Also

D1Client class description.

Examples

## Not run:
library(dataone)
d1c <- D1Client("PROD", "urn:node:KNB")
queryParams <- list(q="id:doi*", rows="5",
   fq="(abstract:chlorophyll AND dateUploaded:[2000-01-01T00:00:00Z TO NOW])",
   fl="title,id,abstract,size,dateUploaded,attributeName")
result <- d1SolrQuery(d1c, queryParams)

## End(Not run)
### d1_errors

This function parses a DataONE service response message for errors, and extracts and prints error information.

#### Description

This function parses a DataONE service response message for errors, and extracts and prints error information.

#### Usage

```r
d1_errors(x)
```

#### Arguments

- `x` The DataONE service response

### data.characterEncoding

**CharacterEncoding**

#### Description

The character encoding used, for example "UTF-8"

#### Usage

```r
data.characterEncoding(x, index, ...)
```

#### Arguments

- `x` the TableDescriber
- `index` index of the table within the document
- `...` Additional parameters

#### Value

the encoding used when serializing the data

#### Author(s)

rnahf
**data.formatFamily**

*Data Format*

**Description**

Get the table format family.

**Usage**

```r
data.formatFamily(x, index, ...)
```

```r
## S4 method for signature 'EMLParser,numeric'
data.formatFamily(x, index)
```

**Arguments**

- `x`: the TableDescriber
- `index`: index of the table within the document
- `...`: Additional parameters

**Value**

the format of the data object being described

**Author(s)**

rnahf

---

**data.tableAttributeNames**

*returns the attribute names*

**Description**

The attribute names are defined in the metadata document for the specified data table

**Usage**

```r
data.tableAttributeNames(x, index, ...)
```

```r
## S4 method for signature 'EMLParser,numeric'
data.tableAttributeNames(x, index)
```
The Attribute (Header) Orientation

Description

Which way do the attribute headers run? Most data has a header row where the attribute names go across "columns", in row in which case, the return value for this method should be "columns."

Usage

data.tableAttributeOrientation(x, index, ...)

## S4 method for signature 'EMLParser,numeric'
data.tableAttributeOrientation(x, index)

Arguments

x - the TableDescriber instance
index - the index of the table within the document
... Additional parameters

Value

legal values are "columns" | "rows"

Note

this is the opposite question from how records are organized!!

Author(s)

rnahf
data.tableAttributeStorageTypes

returns the attributes’ data storage types

Description

The attributes’ data storage types are defined in the metadata document for the specified data table

Usage

data.tableAttributeStorageTypes(x, index, ...)

## S4 method for signature 'EMLParser,numeric'
data.tableAttributeStorageTypes(x, index)

Arguments

x - the TableDescriber instance
index - the index of the table to get results for
... (not yet used)

Value

the data storage types of the attributes

Author(s)

rnahf

data.tableAttributeTypes

returns the attributes’ data types

Description

The attributes’ data types are defined in the metadata document for the specified data table

Usage

data.tableAttributeTypes(x, index, ...)

## S4 method for signature 'EMLParser,numeric'
data.tableAttributeTypes(x, index)
Arguments

\( x \) - the TableDescriber instance

\( \text{index} \) - the index of the table to get results for

\( \ldots \) - (not yet used)

Value

the data types of the attributes

Author(s)

rnahf

---

data.tableFieldDelimiter

Field Delimiter

Description

Get the table field delimiter.

Usage

data.tableFieldDelimiter(x, index, ...)

## S4 method for signature 'EMLParser, numeric'
data.tableFieldDelimiter(x, index)

Arguments

\( x \) - the TableDescriber

\( \text{index} \) - index of the table within the document

\( \ldots \) - Additional parameters

Value

the field delimiter(s) of the data object being described

Author(s)

rnahf
**data.tableMissingValueCodes**

returns missing value codes

**Description**

the missing value codes are defined in the metadata document for the specified data table

**Usage**

```r
data.tableMissingValueCodes(x, index, ...)
```

## S4 method for signature 'EMLParser,numeric'

```r
data.tableMissingValueCodes(x, index)
```

**Arguments**

- `x` - the TableDescriber instance
- `index` - the index of the table to get results for
- `...` - (not yet used)

**Value**

vector of missing value codes

**Author(s)**

rnahf

**data.tableQuoteCharacter**

*Quote Character*

**Description**

Get the table quote character.

**Usage**

```r
data.tableQuoteCharacter(x, index, ...)
```

## S4 method for signature 'EMLParser,numeric'

```r
data.tableQuoteCharacter(x, index)
```
Arguments

x  the TableDescriber
index  index of the table within the document
...  Additional parameters

Value

the quote characters(s) for the data object being described

Author(s)

rnahf

data.tableSkipLinesHeader

Number of lines to skip before reading data

Description

The specified number of lines are skipped.

Usage

data.tableSkipLinesHeader(x, index, ...)

## S4 method for signature 'EMLParser,numeric'
data.tableSkipLinesHeader(x, index)

Arguments

x  - the TableDescriber
index  - the index of the table within the document
...  Additional parameters

Value

the number of lines to skip

Author(s)

rnahf

See Also

help(read.table)
Search, download and upload data to the DataONE network.

Description

The R package dataone provides read/write access to data and metadata from the DataONE network of Member Node data repositories. Member Nodes in DataONE are independent data repositories that have adopted the DataONE services for interoperability, making each of the repositories accessible to client tools such as the DataONE R Client using a standard interface. The DataONE R Client can be used to access data files and to write new data and metadata files to nodes in the DataONE network.

Classes

- **AuthenticationManager**: AuthenticationManager provides methods to validate DataONE authentication.
- **CNode**: A CNode represents a DataONE Coordinating Node and can be used to access its services.
- **D1Client**: The D1Client class contains methods that perform high level dataone tasks.
- **D1Node**: A base class for CNode and MNode.
- **MNode**: MNode provides functions interacting with a DataONE Member Node repository.

Author(s)

Matthew B. Jones (NCEAS) and Peter Slaughter (NCEAS)

See Also

A description of the dataone R package is available with the command: 'vignette("dataone-overview")'.

describeObject

Efficiently get systemmetadata for an object.

Description

This method provides a lighter weight mechanism than getSystemMetadata() for a client to determine basic properties of the referenced object. This operation requires read privileges for the object specified by 'pid', as is granted with a DataONE authentication token or X.509 certificate.

Usage

describeObject(x, ...)

## S4 method for signature 'D1Node'
describeObject(x, pid)
documented.d1Identifiers

Get DataONE identifiers

Description

Get the DataONE identifier associated with each table

Usage

documented.d1Identifiers(x, ...)

## S4 method for signature 'EMLParser'
documented.d1Identifiers(x)

Arguments

x  
  the TableDescriber

...  
  Additional parameters

Value

A list of header elements

See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MNRead.describe

Examples

## Not run:
library(dataone)
mn_uri <- "https://knb.ecoinformatics.org/knb/d1/mn/v1"
mn <- MNode(mn_uri)
vid <- "knb.473.1"
describeObject(mn, vid)
describeObject(mn, "adfadf") # warning message when wrong vid

## End(Not run)
Value

vector of dataONE identifiers

Author(s)

rnahf

---

documented.entityNames

*Get the entity names associated with each table*

Description

The entity names associated with each table are returned.

Usage

documented.entityNames(x, ...)

## S4 method for signature 'EMLParser'
documented.entityNames(x)

Arguments

  x the TableDescriber

  ... Additional parameters

Value

vector of entity names

Author(s)

rnahf
**documented.sizes**  
*Get the sizes of the described data tables.*

**Description**  
Get the table size.

**Usage**  
```r
documented.sizes(x, ...)
```

## S4 method for signature 'EMLParse'

documented.sizes(x)

**Arguments**

- `x`  
  the TableDescriber

- `...`  
  Additional parameters

**Value**

vector of data table sizes (in bytes)

**Author(s)**

rnahf

**downloadCert**  
*Open the CILogon Certificate download page in the default browser.*

**Description**

A convenience method to take you to the CILogon download page: "https://cilogon.org/?skin=DataONE.
Logging into CILogon will allow you to download your X.509 certificate to your local computer.
Typically, the certificate is saved in the default Globus location for certificates (getCertLocation) and once it is there, the 'dataone' package will use the certificate for all authenticated operations. Deleting the certificate file is the equivalent of logging out.

**Usage**

```r
downloadCert(x, ...)
```

## S4 method for signature 'CertificateManager'

downloadCert(x)
downloadObject

**Arguments**

- **x** 
  a CertificateManager instance
- **...** 
  (Not yet used)

---

**downloadObject**   
*Download an object from the DataONE Federation to Disk.*

**Description**

A convenience method to download an object to disk.

**Usage**

```r
downloadObject(x, identifier, ...)  
```

### S4 method for signature 'D1Client'

downloadObject(x, identifier, path = getwd(), check = as.logical(TRUE))

**Arguments**

- **x** 
  A D1Client object.
- **identifier** 
  The identifier of the object to get.
- **...** 
  (Not yet used.)
- **path** 
  (optional) Path to a directory to write object to. The name of the file will be determined from the SystemMetada of the object (see details for more information). The function will fail if a file with the same name already exists in the directory.
- **check** 
  (optional) A logical value, if TRUE check if this object has been obsoleted by another object in DataONE.

**Details**

This method performs multiple underlying calls to the DataONE repository network. CN.resolve() is called to locate the object on one or more repositories, and then each of these is accessed until success at downloading the associated SystemMetadata for the object. The SystemMetadata is used to assign a name to the file that is output to disk. If a fileName is specified in the SystemMetadata, then the file output to disk will be named according to the SystemMetadata fileName. If there is not a specified SystemMetadata fileName, the identifier will be used as the file name output to disk. If the identifier is used as the file name, a file name extension will be determined using the SystemMetadata formatID along with information from CNCore.listFormats(). If the SystemMetadata formatID is "application/octet-stream" no extension will be written.

**Value**

A path where the output file is written to.
See Also

\texttt{D1Client} class description.

Examples

```r
## Not run:
library(dataone)
d1c <- D1Client("PROD", "urn:node:KNB")
pid <- "solson.5.1"
path <- downloadObject(d1c, pid)
## End(Not run)
```

## not run:
library(dataone)
cn <- CNode("STAGING")
creds <- echoCredentials(cn)
print(creds$person$subject)
## End(Not run)

---

**echoCredentials**

*Echo the credentials used to make the call.*

### Description

This method can be used to verify the client certificate is valid and contains the expected information.

### Usage

```r
echoCredentials(x, ...)
```

#### S4 method for signature 'CNode'

```r
echoCredentials(x)
```

### Arguments

- `x` The coordinating node to send the request to.
- `...` (Not yet used)

### Details

The authentication credentials contained in the X.509 certificate or authentication token are sent with the request.

### Value

A list containing authentication info.

### Examples

```r
## Not run:
library(dataone)
cn <- CNode("STAGING")
creds <- echoCredentials(cn)
print(creds$person$subject)
## End(Not run)
```
EMLParser

Description

Construct an EML parser object.

Usage

EMLParser(d1Object, ...)

## S4 method for signature 'D1object'
EMLParser(d1Object)

Arguments

d1Object The D1Object to obtain data from.
...
Additional parameters

EMLParser-class

Handler for Parsing Table Format Details from Metadata

Description

#' Implements methods to provide parsing instructions for asDataFrame.

Details

#' handles eml formats 2.0.0 through 2.1.1

Slots

d1Object the metadata object
xmlDocRoot the xml representation of the metadata

Author(s)

rnahf
encodeSolr  
_Encode the input for Solr Queries_

**Description**

Treating all special characters and spaces as literals, backslash escape special characters, and double-quote if necessary.

**Usage**

```r
encodeSolr(x, ...)
```

## S4 method for signature 'character'

```r
encodeSolr(x, ...)
```

**Arguments**

- `x`: a string to encode
- `...`: (not yet used.)

**Value**

the encoded form of the input

**Examples**

```r
encodeSolr("this & that")
```

---

encodeUrlPath  
_Encode the Input for a URL Path Segment._

**Description**

Encodes the characters of the input so they are not interpreted as reserved characters in url strings. Will also encode non-ASCII unicode characters.

**Usage**

```r
encodeUrlPath(x, ...)
```

## S4 method for signature 'D1Client'

```r
encodeUrlPath(x, pathSegment, ...)
```
encodeUrlQuery

Arguments

x  A D1Client object

... (Not yet used.)

pathSegment : a string to encode

Value

the encoded form of the input

See Also

d1Client class description.

Examples

## Not run:
d1c <- D1Client("STAGING", "urn:node:mnStageUCSB2")
fullyEncodedPath <- paste0("cn/v1/object/",
    encodeUrlPath(d1c, "doi:10.6085/AA/YBHX00_XXXITBDXMMR01_20040720.50.5"))

## End(Not run)

---

encodeUrlQuery | Encode the Input for a URL Query Segment.

Description

Encodes the characters of the input so they are not interpreted as reserved characters in url strings. Will also encode non-ASCII unicode characters.

Usage

encodeUrlQuery(x, ...)

## S4 method for signature 'D1Client'
encodeUrlQuery(x, querySegment, ...)

Arguments

x  A D1Client object.

... (Not yet used.)

querySegment : a string to encode

Value

the encoded form of the input
evaluateAuth

Evaluate DataONE authentication.

Description

A valid DataONE authentication method is looked for and all authentication information is retrieved from it.

Usage

evaluateAuth(.Object, ...)

## S4 method for signature 'AuthenticationManager'
evaluateAuth(.Object, node)

Arguments

,Object 

an Authentication Object.

... additional parameters

node A D1Node object.

Details

If the node specified in the 'node' parameter is a DataONE v2 node or higher, then an authentication token is checked if one exists. If it is readable and not expired, then information for the token is returned. If a valid token does not exist, then the X.509 certificate is checked, if it exists. If it is valid then information is returned for the certificate.

Value

A table containing authentication information.
generateIdentifier

Get a unique identifier that is generated by the Member Node repository and guaranteed to be unique.

Description

Creating objects requires use of a unique persistent identifier (pid) when calling the create function. Member Nodes may optionally provide the generateIdentifier service to issue such identifiers, ensuring that they are unique. Each identifier conforms to an identifier scheme, which determines the syntax and rules for how the identifier that is generated is formatted. All Member Nodes that implement this method must support the UUID scheme, but may also support other schemes such as DOI and others.

Usage

generateIdentifier(x, ...)

## S4 method for signature 'MNode'
generateIdentifier(x, scheme = "UUID", fragment = NULL)

Arguments

- `x` The MNode instance on which the object will be created
- `...` (Not yet used.)
- `scheme` The identifier scheme to be used, such as DOI, UUID, etc.
- `fragment` An optional fragment to be prepended to the identifier for schemes that support it (not all do).

Details

In the version 2.0 library and higher, this operation can utilize an `dataone_token` option to provide credentials for write operations in DataONE. The authentication token is obtained from DataONE (see your profile on https://search.dataone.org). See the vignette("dataone-overview") for details. Alternatively, the version 1.0 approach of using an X.509 certificate in a default location of the file system can also be used. This certificate provides authentication credentials from CILogon https://cilogon.org/?skin=DataONE. See vignette("dataone-overview") for details.

Value

the character string of the generated unique identifier

See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MNStorage.generateIdentifier
getAuthMethod

Examples

```r
## Not run:
library(dataone)
cn <- CNode("STAGING")
mn <- getMNode(cn, "urn:node:mmStageUCSB2")
newid <- generateIdentifier(mn, "UUID")

## End(Not run)
```

getAuthExpires  \hspace{1cm} \textit{Get the expiration date of the current authentication method.}

Description

The expiration date of the current authentication method, either authentication token or X.509 certificate, is returned as a Greenwich Mean Time (GMT) value.

Usage

```r
getAuthExpires(.Object, node)
```

## S4 method for signature 'AuthenticationManager'
```r
getAuthExpires(.Object, node)
```

Arguments

- `.Object`: An AuthenticationManager instance
- `node`: A D1Node instance

Value

The expiration date for the current authentication mechanism being used.

gAuthMethod  \hspace{1cm} \textit{Get the current valid authentication mechanism.}

Description

Get the current valid authentication mechanism.

Usage

```r
gAuthMethod(.Object, ...)
```

## S4 method for signature 'AuthenticationManager'
```r
gAuthMethod(.Object, node)
```
getAuthSubject

Arguments

,Object An AuthenticationManager instance
... (Not yet used)
node A D1Node instance to determine the authentication method for.

Details

The current authentication method being used, either an authentication token or an X.509 certificate. The 'node' argument is used to determine the authentication mechanism that is appropriate for the specified 'node'. For example, authentication tokens are supported on DataONE nodes that use the DataONE V2.0 API or higher, so if the node uses the V1 API, then only an X.509 certificate can be used.

Value

The current authentication mechanism as a character string, either "token" or "cert".

getAuthSubject Get the authentication subject.

Description

Get the authentication subject.

Usage

getAuthSubject(.Object, ...)

## S4 method for signature 'AuthenticationManager'
getAuthSubject(.Object, node)

Arguments

.Object an AuthenticationManager instance
... (Not yet used)
node A D1Node instance

Details

The authenticated user, aka 'subject' is retrieved from the authentication mechanism currently being used, either an authentication token or an X.509 certificate. The 'node' argument is used to determine the authentication mechanism that is appropriate for the specified 'node'. For example, authentication tokens are supported on DataONE nodes that use the DataONE V2.0 API or higher, so if the node uses the V1 API, then only an X.509 certificate can be used.

Value

the DataONE Subject that is your client’s identity
getCapabilities

Get the node capabilities description, and store the information in the MNode.

Description

Access the DataONE getCapabilities() service for the Member Node, which returns an XML description of the repository and the services it offers.

Usage

getCapabilities(x, ...)

## S4 method for signature 'MNode'
getCapabilities(x)

Arguments

x The node identifier with which this node is registered in DataONE

... (Not yet used.)

Value

an XMLInternalDocument object representing the DataONE environment

See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MN_core.getCapabilities

Examples

## Not run:
library(dataone)
cn <- CNode()
mn <- getMNode(cn, "urn:node:KNB")
xml <- getCapabilities(mn)

## End(Not run)
**getCert**  

Get the DataONE X.509 Certificate location.

**Description**

Get the DataONE X.509 Certificate location.

**Usage**

```
getCert(.Object, ...)
```

## S4 method for signature 'AuthenticationManager'

```
getCert(.Object)
```

**Arguments**

- `.Object` an AuthenticationManager instance
- `...` (Not yet used)

**Value**

The filename of the current DataONE X.509 Certificate if it is available.

---

**getCertExpires**  

Show the date and time when an X.509 certificate expires.

**Description**

Each X.509 has a range of certificate validity times. This method returns the X.509 `notAfter` field formatted as a `POSIXct` date value.

**Usage**

```
getCertExpires(x, ...)
```

## S4 method for signature 'CertificateManager'

```
getCertExpires(x)
```

**Arguments**

- `x` a CertificateManager instance
- `...` (Not yet used)

**Value**

` POSIXct ` value
getCertInfo

Get X.509 Certificate information

Description

The DataONE X.509 certificate is read, if it is present and the information contained in the certificate is returned as a data.frame.

Usage

getCertInfo(.Object)

## S4 method for signature 'AuthenticationManager'
getCertInfo(.Object)

Arguments

/Object an Authentication Object.

Value

A data.frame containing information about the X.509 certificate.

getCertLocation

Get the file path on disk of the client certificate file.

Description

Find the location of the client certificate, which is typically in a default location on disk, unless the 'location' slot has been set with a custom location for the certificate.

Usage

getcertLocation(x, ...)

## S4 method for signature 'CertificateManager'
getcertLocation(x)

Arguments

x a CertificateManager instance
... (Not yet used)
getChecksum

Details

The default Globus Grid Security Infrastructure (GSI) location is './tmp/x509up_u$UID' on Unix or '${tmpdir}/x509up_u$UID' on Windows or '${tmpdir}/x509up_u$user.name' if '${UID}' is not defined.

Value

character the path to the certificate

getChecksum | Get the checksum for the data object associated with the specified pid.

Description

A checksum is calculated for an object when it is uploaded to DataONE and is submitted with the object's system metadata. The 'getChecksum' method retrieves the checksum from the specified coordinating node

Usage

getChecksum(x, ...)

## S4 method for signature 'CNode'
getChecksum(x, pid, ...)

## S4 method for signature 'MNode'
getChecksum(x, pid, checksumAlgorithm = "SHA-256")

Arguments

x The CNode instance from which the checksum will be retrieved
... (Not yet used)
pid The identifier of the object
checksumAlgorithm The algorithm used to calculate the checksum. Default="SHA-256"

Value

character the checksum value, with the checksum algorithm as the attribute "algorithm"

See Also

D1Node-class{D1Node} class description.
getCN

Get the coordinating node associated with this D1Client object.

Description

Get the coordinating node associated with this D1Client object.

Usage

getCN(x)

## S4 method for signature 'D1Client'
getCN(x)

Arguments

x A D1Client object.

Note

The method getCN has been deprecated.

See Also

D1Client class description.

Examples

## Not run:
cli <- D1Client("STAGING2", "urn:node:mnTestKNB")
testCN <- getCN(cli)

## End(Not run)
getD1Object

Download a data object from the DataONE Federation.

Description
An object is download from the DataONE network for the identifier that is provided.

Usage
getD1Object(x, identifier, ...)

## S4 method for signature 'D1Client'
getD1Object(x, identifier)

Arguments

x A D1Client instance
identifier The identifier of the object to download from DataONE
... (not yet used)

Value
A datapack:DataObject

See Also
D1Client class description.

Examples
## Not run:
library(dataone)
d1c <- D1Client("PROD", "urn:node:KNB")
pid <- "solson.5.1"
dataObj <- getD1Object(d1c, pid)
data <- getData(dataObj)
## End(Not run)
### `getDataObject`

#### Description

A convenience method to download a data object and its associated SystemMetadata, wrapped in a `DataObject` class.

#### Usage

```r
getDataObject(x, identifier, ...)  

## S4 method for signature 'D1Client'

getDataObject(  
  x,  
  identifier,  
  lazyLoad = FALSE,  
  limit = "1MB",  
  quiet = TRUE,  
  checksumAlgorithm = as.character(NA)
)
```

#### Arguments

- `x`: D1Object the data structure from where to get the data
**Arguments**

- **x**
  - A D1Client object.
- **identifier**
  - The identifier of the object to get.
- **lazyLoad**
  - A logical value. If TRUE, then only package member system metadata is downloaded and not data.
- **limit**
  - A character value specifying maximum package member size to download. Specified with "KB", "MB" or "TB" for example: "100KB", "10MB", "20GB", "1TB". The default is "1MB". Only takes effect if lazyLoad=FALSE.
- **quiet**
  - A logical. If TRUE (the default) then informational messages will not be printed.
- **checksumAlgorithm**
  - A character value specifying the algorithm to use to re-calculate (after download) the system metadata checksum for the object’s data bytes for example: "SHA-256". The default is "NA", which specifies that this re-calculation will not be performed.

**Details**

This method performs multiple underlying calls to the DataONE repository network. CN.resolve() is called to locate the object on one or more repositories, and then each of these is accessed until the associated SystemMetadata and data bytes are successfully downloaded. This data is then used to construct the returned DataObject. This function replaces the previous getD1Object() method in the version 1 dataone library.

The `lazyLoad` parameter controls whether the data bytes for a DataONE item are downloaded (the system metadata is always downloaded). When `lazyLoad=FALSE`, the `limit` parameter can be used to specify the maximum size of a data file that will be downloaded. If `lazyLoad` is TRUE, then `limit` is ignored. The `lazyLoad` and `limit` can be used together in the following ways:

<table>
<thead>
<tr>
<th>'lazyLoad'</th>
<th>'limit'</th>
<th>result</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Any value</td>
<td>Data bytes are not downloaded</td>
<td>The 'limit' parameter is ignored</td>
</tr>
<tr>
<td>FALSE</td>
<td>Not specified</td>
<td>Data bytes are downloaded if less than 1MB</td>
<td>The default 'limit' of 1MB is used</td>
</tr>
<tr>
<td>FALSE</td>
<td>10MB</td>
<td>Data bytes are downloaded if less than 10MB</td>
<td>The specified 'limit' values is used</td>
</tr>
</tbody>
</table>

Note that DataONE system metadata is always downloaded and populated into the resulting DataObject, regardless of the 'lazyLoad' and 'limit' values specified in the call to `getDataObject()`.

**Value**

A DataObject or NULL if the object was not found in DataONE

**See Also**

D1Client class description.
getDataPackage

Download data from the DataONE Federation as a DataPackage.

Description

This is a convenience method that will download all the members in a DataONE data package and insert them into a DataPackage, including associated SystemMetadata for each package member.

Usage

getDataPackage(x, identifier, ...)

## S4 method for signature 'D1Client'
getDataPackage(
  x,
  identifier,
  lazyLoad = FALSE,
  limit = "1MB",
  quiet = TRUE,
  checksumAlgorithm = as.character(NA)
)

Arguments

x
A D1Client object.

identifier
The identifier of a package, package metadata or other package member

... (not yet used)

lazyLoad
A logical value. If TRUE, then only package member system metadata is downloaded and not data. The default is FALSE.

limit
A character value specifying maximum package member size to download. Specified with "KB", "MB" or "TB" for example: "100KB", "10MB", "20GB", "1TB". The default is "1MB". Only takes effect if 'lazyLoad=FALSE'.

quiet
A 'logical'. If TRUE (the default) then informational messages will not be printed.

Examples

## Not run:
library(dataone)
d1c <- D1Client("PROD", "urn:node:KNB")
pid <- "solson.5.1"
obj <- getDataObject(d1c, pid)
data <- getData(obj)

## End(Not run)
getEndpoint

checksumAlgorithm
A character value specifying the algorithm to use to re-calculate (after download) the system metadata checksum for the object's data bytes for example: "SHA-256". The default is "NA", which specifies that this re-calculation will not be performed.

Details
A ‘data package’ that resides on a DataONE member node is defined as a collection of digital objects that are described by a metadata document.

The lazyLoad parameter controls whether the data bytes for a DataONE package member are downloaded (the system metadata is always downloaded). When lazyLoad=FALSE, the limit parameter can be used to specify the maximum size of a data file that will be downloaded. If lazyLoad is TRUE, then limit is ignored. The lazyLoad and limit parameters can be used together in the following ways:

<table>
<thead>
<tr>
<th>lazyLoad</th>
<th>limit</th>
<th>result</th>
<th>comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>Any value</td>
<td>Data bytes are not downloaded</td>
<td>The 'limit' parameter is ignored</td>
</tr>
<tr>
<td>FALSE</td>
<td>Not specified</td>
<td>Data bytes are downloaded if less than 1MB</td>
<td>The default 'limit' of 1MB is used</td>
</tr>
<tr>
<td>FALSE</td>
<td>10MB</td>
<td>Data bytes are downloaded if less than 10MB</td>
<td>The specified 'limit' values is used</td>
</tr>
</tbody>
</table>

Value
A DataPackage or NULL if the package was not found in DataONE

See Also
D1Client class description.

Examples
```r
## Not run:
library(dataone)
de <- D1Client("PROD", "urn:node:KNC")
pid <- "solson.5.1"
pp <- getDataPackage(dlc, pid)
## End(Not run)
```

getEndpoint
Return the URL endpoint for the DataONE Coordinating Node.

Description
A D1Client object is associated with a DataONE Coordinating Node. This CN is either the production CN (from the "PROD" environment, the default), or a CN from one of the development environments ("STAGING", "SANDBOX", "DEV"). The base URL for the CN is returned.
Usage

getEndpoint(x, ...)

## S4 method for signature 'D1Client'
getEndpoint(x)

Arguments

x A D1Client object

... (Not yet used.)

Value

A character vector containing the URL of the Coordinating Node

See Also

D1Client class description.

Examples

## Not run:
cli <- D1Client("STAGING2", "urn:node:mnTestKNB")
cnUrl <- getEndpoint(cli)

## End(Not run)

gErrorDescription

Extract an error message from an http response.

Description

Http requests can fail for a variety of reasons, so getErrorDescription first tries to determine what type of response was sent.

Usage

gErrorDescription(response)

Arguments

response The httr response object to extract the error description from.
getFormat

Details

The return types handled by this function are:

- An incorrect url is sent to DataONE and an error is returned by the web server, not a specified DataONE service url. In this case, a generic error message may be returned, e.g. status=404, URL not found
- A DataONE service was called, and returned an error message. In this case the DataONE response is parsed in an attempt to retrieve a meaningful error message.

getFormat | Get information for a single DataONE object format

Description

Get information for a single DataONE object format

Usage

getFormat(x, ...)

## S4 method for signature 'CNode'
getFormat(x, formatId)

Arguments

x A CNode object instance
...
(Not yet used)
formatId The formatId to retrieve.

Value

A dataframe of all object formats registered in the DataONE Object Format Vocabulary.

See Also

CNode class description.

Examples

## Not run:
library(dataone)
cn <- CNode()
fmt <- getFormat(cn, "eml://ecoinformatics.org/eml-2.1.0")
cat(sprintf("format name: %s\n", fmt$name))
cat(sprintf("format type: %s\n", fmt$type))
cat(sprintf("Format Id: %s\n", fmt$id))

## End(Not run)
## getFormatId, D1Object-method

*Get the FormatId of the D1Object*

### Description

Get the FormatId of the D1Object

### Usage

```r
## S4 method for signature 'D1object'
getFormatId(x)
```

### Arguments

- `x` - D1Object

### Value

the formatId

## getIdentifier, D1Object-method

*Get the Identifier of the D1Object*

### Description

Get the Identifier of the D1Object

### Usage

```r
## S4 method for signature 'D1object'
getIdentifier(x)
```

### Arguments

- `x` - D1Object

### Value

the identifier
getMetadataMember

Get the DataObject containing package metadata

Description

Each DataObject in the DataPackage is inspected to see if it matches one of the formats supported by DataONE for metadata. If a package member’s format matches one of the supported formats, the identifier for that member is returned.

Usage

getMetadataMember(x, dp, ...)

## S4 method for signature 'D1Client,DataPackage'
getMetadataMember(x, dp, as = "character", ...)

Arguments

x   A D1Client object
dp   A DataPackage object
...  (Additional arguments, Not yet used.)
as   A value of type "character" that specifies the return value. Possible values are "character" (the default) or "DataPackage".

Details

This method calls the DataONE CN 'format' service to obtain the current format list.

Value

The identifier of the metadata object

getMN

Get a member node client based on its node identifier.

Description

Get a member node client based on its node identifier.
**getMNode**

Get a reference to a node based on its identifier

**Usage**

getMNode(x, ...)   

## S4 method for signature 'D1Client,ANY'
getMNode(x, nodeid, ...)

## S4 method for signature 'D1Client,character'
getMNode(x, nodeid)

**Arguments**

x                  A D1Client object.
nodeid             The identifier of the node to retrieve.
...                (Not yet used)

**Note**

This method has been superceded by `getMNodeId`

**See Also**

`D1Client` class description.

**Examples**

## Not run:
cli <- D1Client("STAGING2", "urn:node:mnTestKNB")
testMN <- getMNode(cli)

## End(Not run)

---

**getMNode**

Get a reference to a node based on its identifier

**Description**

Get a reference to a node based on its identifier

**Usage**

getMNode(x, ...)

## S4 method for signature 'CNode'
getMNode(x, nodeid)
getMNodeId

Arguments

x The coordinating node to query for its registered Member Nodes
...
nodeid The standard identifier string for this node

Details

For an explanation of DataONE Coordinating Nodes and Member Node identifiers, see the section "DataONE Environments" in the overview vignette by entering the R command: vignette("dataone-overview").

Value

the Member Node as an MNode reference, or NULL if not found

See Also

CNode class description.

Examples

## Not run:
cn <- CNode()
mn <- getMNode(cn, "urn:node:KNB")
## End(Not run)

getMNodeId

Get the member node identifier associated with this D1Client object.

Description

One Member Node can be associated with the client as the default to which data and metadata are written.

Usage

getMNodeId(x)

## S4 method for signature 'D1Client'
getMNodeId(x)

Arguments

x A D1Client object.

Value

The Member Node identifier as a character vector
See Also

D1Client class description.

Examples

```r
## Not run:
cli <- D1Client("STAGING2", "urn:node:mnTestKNB")
mn <- getMNodeId(cli)

## End(Not run)
```

### Description

Get the bytes associated with an object on this Node.

### Usage

```r
generate(x, ...)  
## S4 method for signature 'CNode'
generate(x, pid)  
## S4 method for signature 'MNode'
generate(x, pid, check = as.logical(FALSE))
```

#### Arguments

- **x**: The Node instance from which the pid will be downloaded
- **...**: (Not yet used).
- **pid**: The identifier of the object to be downloaded
- **check**: A logical value, if TRUE check if this object has been obsoleted by another object in DataONE.

#### Details

This operation acts as the 'public' anonymous user unless an X.509 certificate is present in the default location of the file system, in which case the access will be authenticated.

#### Value

the bytes of the object

See Also

D1Node-class{D1Node} class description.
getPackage

Examples

```r
## Not run:
library(dataone)
cn <- CNode()
mn <- getMNode(cn, "urn:node:KNB")
pid <- "solson.5.1"
obj <- getObject(mn, pid)
df <- read.csv(text=rawToChar(obj))

## End(Not run)
```

getPackage  Download a data package from a member node.

Description

Given a valid identifier, download a file containing all of the package members of the corresponding DataONE data package.

Usage

```r
getPackage(x, ...)
```

## S4 method for signature 'MNode'
```r
getPackage(
  x,
  identifier,
  format = "application/bagit-097",
  dirPath = NULL,
  unzip = FALSE
)
```

Arguments

- `x`  A MNode instance representing a DataONE Member Node repository.
- `...` (not yet used)
- `identifier`  The identifier of the package to retrieve. The identifier can be for the resource map, metadata file, data file, or any other package member.
- `format`  The format to send the package in.
- `dirPath`  The directory path to save the package to.
- `unzip`  (logical) If the dirPath is specified, the package can also be unzipped automatically (unzip=TRUE).
Details
The default data package file format is a Bagit file (https://tools.ietf.org/html/draft-kunze-bagit-09). The downloaded package file is compressed using the ZIP format and will be located in an R session temporary file. Other packaging formats can be requested if they have been implemented by the requested member node.

Value
The location of the package file downloaded from the member node.

See Also
MNode class description.

Examples
```R
## Not run:
 cn <- CNode()
 mn <- getMNode(cn, "urn:node:KNB")
 packageFileName <- getPackage(mn, id="resourceMap_Blandy.76.2")

## End(Not run)
```

---

**getQueryEngineDescription**

_QUERY a node for the list of query engines available on the node_

Description
Query a node for the list of query engines available on the node

Usage
```
getQueryEngineDescription(x, ...)
```

## S4 method for signature 'D1Node'

```
getQueryEngineDescription(x, queryEngineName)
```

Arguments
- `x` The CNode or MNode to query
- `...` (Additional arguments - not yet used.)
- `queryEngineName` The query engine name to get a description for.

Value
- `list` The query engine description
getSystemMetadata

Examples

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

# Get the query engine description
cn <- CNode("PROD")
engineDesc <- getQueryEngineDescription(cn, "solr")
cat(sprintf("Query engine version: %s\n", engineDesc$queryEngineVersion))
cat(sprintf("Query engine name: %s\n", engineDesc$name))

# Get the query engine description again
engineDesc <- getQueryEngineDescription(cn, "solr")
head(engineDesc$queryFields, n=3L)

## End(Not run)
```

getSystemMetadata Get the metadata describing system properties associated with an object on this Node.

Description

The SystemMetadata includes information about the identity, type, access control, and other system level details about the object.

Arguments

- `x`: The Node instance from which the SystemMetadata will be downloaded
- `pid`: The identifier of the object
- `...`: (Not yet used.)

Details

This operation acts as the 'public' anonymous user unless an X.509 certificate is present in the default location of the file system, in which case the access will be authenticated.
getToken

Get the value of the DataONE Authentication Token, if one exists.

Description

Get the value of the DataONE Authentication Token, if one exists.

Usage

getToken(.Object, ...)

## S4 method for signature 'AuthenticationManager'

getMethod(.Object, node = as.character(NA))

Arguments

/Object/ an AuthenticationManager instance
... additional parameters
/node/ either a CNode or MNode object to get the appropriate token for.
**Details**

A token value is retrieved based on the DataONE environment that the specified node is located in, either the production environment or a test environment.

**Value**

The current authentication token.

---

**getTokenInfo**  
*Get authentication token information*

---

**Description**

The DataONE authentication token is read, if it has been set, and the information it contains is returned as a data.frame.

**Usage**

```
getTokenInfo(.Object)
```

### S4 method for signature 'AuthenticationManager'

```
getTokenInfo(.Object)
```

**Arguments**

- `.Object` an Authentication Object.

**Value**

A data.frame containing information about the authentication token.

---

**get_user_agent**  
*User agent string*

---

**Description**

Get a string representation of the user agent to be sent to the server along with other request details.

**Usage**

```
get_user_agent()
```
hasReservation

Checks to determine if the supplied subject is the owner of the reservation of id.

Description

The hasReservation method checks the reservation of an identifier that has previously been reserved with the reserveIdentifier method. The identifier must have been reserved by the specified DataONE user identity (subject).

Usage

hasReservation(x, ...)

## S4 method for signature 'CNode'

hasReservation(x, pid, subject = as.character(NA))

Arguments

x
A CNode instance.

... Additional parameters.

pid
The identifier that is being checked for existing as a reserved identifier or is in use as an identifier for an existing object

subject
The subject of the principal (user) that made the reservation.

Details

To determine the DataONE identity that is currently being used for DataONE authentication, use the echoCredentials method.

Value

A logical value where TRUE means a reservation exists for the specified pid by the subject.

See Also

CNode class description.

Examples

## Not run:
library(dataone)
cn <- CNode("STAGING")
creds <- echoCredentials(cn)
subject <- creds$person$subject
# Previously reserved pid (using reserveIdentifier()), e.g. DOI or uuid
pid <- "urn:node:e27bb4f3-96bb-4af4-8902-f5914def077c"
hasRes <- hasReservation(cn, pid, subject=subject)

## End(Not run)

### initialize,D1Client-method

**Initialize a D1Client object**

#### Description

Initialize a D1Client object

#### Usage

```r
## S4 method for signature 'D1Client'
initialize(
  .Object,
  cn = NA,
  mn = NA,
  env = as.character(NA),
  mNodeid = as.character(NA)
)
```

#### Arguments

- `.Object` A D1client object.
- `cn` The Member Node object to associate this D1Client with.
- `mn` The Member Node object to associate this D1Client with.
- `env` The DataONE environment to initialize this D1Client with, e.g. "PROD", "STAGING", "SANDBOX", "DEV"
- `mNodeid` The node identifier of the Member Node to associate with this D1Client.

#### See Also

dataone class description.

#### Examples

```r
## Not run:
library(dataone)
d1c <- D1Client("PROD", "urn:node:KNB")

## End(Not run)
```
initialize,D1Node-method

Initialize a D1Node

Description

Initialize a D1Node

Usage

## S4 method for signature 'D1Node'
initialize(.Object)

Arguments

<Object> the D1Node object

initialize,D1Object-method

Initialize a D1Object

Description

Initialize a D1Object

Usage

## S4 method for signature 'D1Object'
initialize(.Object, id, data, format, mnNodeId = as.character(NA))

Arguments

<Object> A D1Object instance.
$id$ The identifier for the object.
$data$ An R object (data or metadata) that this D1Object contains.
$format$ The Object format.
$mnNodeId$ The DataONE node identifier associated with this object, i.e. "urn:node:KNB"

See Also

D1Object class description.
isAuthExpired

Check if the currently valid authentication method has reached the expiration time.

Description

Check if the currently valid authentication method has reached the expiration time.

Usage

isAuthExpired(.Object, ...)

## S4 method for signature 'AuthenticationManager'

isAuthExpired(.Object, node)

Arguments

/Object/ An AuthenticationManager instance

... (Not yet used)

node A D1Node instance

Value

A logical value: TRUE if authentication has expired, FALSE if not.

isAuthorized

Check if an action is authorized for the specified identifier

Description

Test if the user identified by the provided token has authorization for operation on the specified object.

Usage

isAuthorized(x, ...)

## S4 method for signature 'D1Node'

isAuthorized(x, id, action)

Arguments

x The node to send the request to. This is either a "CNode" or "MNode" instance.

... (Not yet used)

id The DataONE identifier (pid or sid) to check access for.

action The DataONE action to check, possible values: "read", "write", "changePermission"
Details

The identifier parameter may be either a DataONE persistent identifier (pid) or series identifier (sid).

Value

a logical, TRUE if the action is authorized, false if not.

See Also

CNode class description.

Examples

## Not run:
# Send an authorization check to the D1 production CN.
cn <- CNode("PROD")
pid <- "doi:10.6073/pasta/7fcb8feda57843fae65f63094472f502d"
canRead <- isAuthorized(cn, pid, "read")
canWrite <- isAuthorized(cn, pid, "write")
canChange <- isAuthorized(cn, pid, "changePermission")

# Now send a check to a member node.
mn <- getMNode(cn, "urn:node:KNB")
pid <- "doi:10.6085/AA/pisco_recruitment.149.1"
canRead <- isAuthorized(mn, pid, "read")
canWrite <- isAuthorized(mn, pid, "write")
canChange <- is Authorized(mn, pid, "changePermission")

## End(Not run)

---

isAuthValid Verify authentication for a member node.

Description

The currently used DataONE client authentication method (either tokens or X.509 certificates) is checked and verified for the specified node (either CN or MN). If an authentication token is available via the R options facility, it will be used i.e. available via getOption("dataone_token"). However, authentication tokens can only be used for DataONE v2 or higher nodes. X.509 certificates can be used with DataONE v1 or higher nodes. See the "dataone" vignette "dataone-overview" for more information on authentication.

Usage

isAuthValid(.Object, ...)

## S4 method for signature 'AuthenticationManager'
isAuthValid(.Object, node)
isCertExpired

Arguments

= Object
  An AuthenticationManager instance

... additional parameters

= node
  The node object (MNode or CNode) that authentication is being checked for.

Value

A logical value: TRUE if authentication is valid, false if not.

isCertExpired = Determine if an X.509 certificate has expired.

Description

Returns 'TRUE' if the certificate associated with a CertificateManager instance is expired. A certificate is expired if any of the following conditions hold: 1) the current time is before or after the certificate validity dates, 2) the certificate is not valid according to a trusted Certificate Authority, or 3) no certificate can be found.

Usage

isCertExpired(x, ...)

## S4 method for signature 'CertificateManager'
isCertExpired(x)

Arguments

= x
  a CertificateManager instance

... (Not yet used)

Value

TRUE if the certificate is expired
listFormats List all object formats registered in DataONE.

Description

The listFormats method queries a DataONE Coordinating Node for a list of all entries in the Object Format Vocabulary.

Usage

listFormats(x, ...)

## S4 method for signature 'CNode'
listFormats(x)

Arguments

x a valid CNode object
...
(Not yet used)

Value

Returns a dataframe of all object formats registered in the DataONE Object Format Vocabulary.

See Also

CNode class description.

Examples

## Not run:
library(dataone)
cn <- CNode()
formats <- listFormats(cn)

## End(Not run)

listMemberNodes List DataONE Member Nodes.

Description

A D1Client object is associated with a DataONE Coordinating Node. The listMemberNodes method lists all member nodes associated with a CN.
listNodes

Get the list of nodes associated with a CN

Usage

listMemberNodes(x)

## S4 method for signature 'D1Client'
listMemberNodes(x)

Arguments

x

A D1Client object.

See Also

D1Client class description.

Examples

## Not run:
d1c <- D1Client("PROD")
nodelist <- listMemberNodes(d1c)

## End(Not run)


d1c <- D1Client("PROD")
nodelist <- listMemberNodes(d1c)

## End(Not run)


Description

Get the list of nodes associated with a CN

Usage

listNodes(x, ...)

## S4 method for signature 'CNode'
listNodes(x, url = as.character(NA), ...)

Arguments

x

The coordinating node to query for its registered Member Nodes

... (Not yet used)

url

Optional - the url of the CN.

Value

the list of nodes in the DataONE CN environment
See Also

CNode class description.

Examples

```r
## Not run:
cn <- CNode()
nodelist <- listNodes(cn)
nodeid <- nodelist[[2]]@identifier

## End(Not run)
```

### listObjects

Retrieve the list of objects that match the search parameters

#### Description

Retrieve the list of objects that match the search parameters

#### Usage

```r
listObjects(x, ...)
```

#### Arguments

- `x` The Node instance from which the SystemMetadata will be downloaded
- `...` (Not yet used.)
- `fromDate` Entries with a modified date greater than 'fromDate' will be returned. This value must be specified in ISO 8601 format, i.e. "YYYY-MM-DDTHH:MM:SS.mmm+00:00"
- `toDate` Entries with a modified date less than 'toDate' will be returned. This value must be specified in ISO 8601 format, i.e. "YYYY-MM-DDTHH:MM:SS.mmm+00:00"
- `formatId` The format to match, for example "eml://ecoinformatics.org/eml-2.1.1"
- `replicaStatus` A logical value that determines if replica (object not on it's origin node) should be returned. Default is TRUE.
- `start` An integer that specifies the first element of the result set that will be returned
- `count` An integer that specifies how many results will be returned
Details

The list of objects that is returned is paged according to the `start` and `count` values, so that large result sets can be returned over multiple calls.

Value

list Objects that met the search criteria

See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MN_read.listObjects

Examples

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

cn <- CNode("STAGING")

fromDate <- "2013-01-01T01:01:01.000+00:00"

toDate <- "2015-12-31T01:01:01.000+00:00"

formatId <- "eml://ecoinformatics.org/eml-2.1.0"

start <- 0

count <- 5

objects <- listObjects(cn, fromDate=fromDate, toDate=toDate,
                        formatId=formatId, start=start, count=count)

# Inspect id of first object

objects[[1]]$objectInfo$identifier

## End(Not run)
```

### listQueryEngines

**Query a node for the list of query engines available on the node**

**Description**

Query a node for the list of query engines available on the node

**Usage**

`listQueryEngines(x, ...)`

```r
## S4 method for signature 'D1Node'

listQueryEngines(x)
```

**Arguments**

- `x` The CNode or MNode to list the query engines for.
- `...` (Not yet used.)
MNode

**Value**

list The list of query engines.

**Examples**

```r
## Not run:
cn <- CNode("STAGING")
engines <- listQueryEngines(cn)
## End(Not run)
```

---

### MNode

Create a MNode object representing a DataONE Member Node repository.

**Description**

Construct an instance of MNode to provide mechanisms to access, create, and update data and metadata objects on the associated Member Node.

**Usage**

```r
MNode(x)
```

*## S4 method for signature 'character'*

```r
MNode(x)
```

*## S4 method for signature 'D1Node'*

```r
MNode(x)
```

**Arguments**

- `x` a URI representing a base URL (i.e. https://knb.ecoinformatics.org/knb/d1/mn/v2); or a reference to a dataone::Node instance

**Details**

If the `x` is a string, it is treated as a URI and an attempt to find an associated Member Node at that base URL is attempted. If `x` is a Node reference, then it is cast to a MNode instance. This typically is used from the getMNode() function from the CNode class, which is the preferred way to retrieve an instance of an MNode.

**Value**

the MNode object-

**See Also**

MNode class description.
Examples

```r
## Not run:
mn <- MNode("https://knb.ecoinformatics.org/knb/d1/mn/v2")

## End(Not run)
```

## Description

MNode provides functions that interact with a DataONE Member Node (MN). A MN is a repository that provides access for reading and writing data and metadata using the DataONE MN service API. The MN API includes functions for retrieving data and metadata based on its unique persistent identifier (pid), as well as for creating, updating, and archiving these data and metadata objects.

## Details

Methods that perform write operations (such as createObject and updateObject) on the MN generally require authentication. For MNs that have implemented the DataONE API version 2.0 and higher, these operations can utilize an authentication token to provide credentials for write operations in DataONE. The authentication token is obtained from DataONE (see your account profile on https://search.dataone.org). See the vignette("dataone-overview") for details. Alternatively, the version 1.0 approach of using an X.509 certificate in a default location of the file system can also be used. This certificate provides authentication credentials from CILogon https://cilogon.org/?skin=DataONE.

## Slots

- **endpoint**: The url to access node services, which is the baseURL plus the version string

## Methods

- **MNode**: Create a MNode object representing a DataONE Member Node repository.
- **createObject**: Create an object on a Member Node.
- **getObject**: Get the bytes associated with an object on the Member Node
- **getCapabilities**: Get the node capabilities description, and store the information in the MNode.
- **generateIdentifier**: Get a unique identifier that is generated by the Member Node repository and guaranteed to be unique.
- **getPackage**: Download a data package from a member node.
- **updateObject**: Update an object to a Member Node, by creating a new object that replaces an original.
- **updateSystemMetadata**: Update the system metadata associated with an object.
See Also

dataone package description.

Examples

```r
## Not run:
library(dataone)
library(uuid)
library(digest)

cn <- CNode("STAGING")
mn <- getMNode(cn, "urn:node:mnStageUCSB2")

# Have Dataone create an identifier for you (requires authentication)
\dontrun{
  newid <- generateIdentifier(mn, "UUID")
}

# Create an identifier manually
newid <- paste("urn:uuid:", UUIDgenerate(), sep="")
testdf <- data.frame(x=1:10, y=11:20)
csvfile <- paste(tempfile(), ".csv", sep="")
write.csv(testdf, csvfile, row.names=FALSE)
f <- "text/csv"

size <- file.info(csvfile)$size
sha256 <- digest(csvfile, algo="sha256", serialize=FALSE, file=TRUE)

sysmeta <- new("SystemMetadata", identifier=newid, formatId=f, size=size,
               checksum=sha256, originMemberNode=mnid, authoritativeMemberNode=mnid)

# Upload data to DataONE (requires authentication)
\dontrun{
  response <- createObject(mn, newid, csvfile, sysmeta)
}
```

## End(Not run)

### obscureAuth

Temporarily disable DataONE authentication.

Description

Calling obscureAuth temporarily disables authentication so that

Usage

```r
obscureAuth(.Object)
```

## S4 method for signature 'AuthenticationManager'

```r
obscureAuth(.Object)
```

Arguments

| .Object | An AuthenticationManager instance |
Details

This method is intended to be used for authentication testing. `isAuthValid` will return FALSE. Authentication can be re-enabled by calling `restoreAuth`.

Value

The expiration date for the current authentication mechanism being used.

---

### obscureCert

**Obscure the CILogon Client Certificate**

Description

Obscures the x509 certificate that CILogon installs, effectively making future interactions with the DataONE services public/anonymous. This function simply renames an existing certificate file to a known location, allowing 'public' operations. Note, when the client certificate is obscured via the renaming, you will not be able to create objects in DataONE, or utilize any other methods that require authentication.

Usage

```r
obscureCert(x, ...)
```

## S4 method for signature 'CertificateManager'

```r
obscureCert(x)
```

Arguments

- `x` a CertificateManager instance
- `...` (Not yet used)

Value

the modified CertificateManager instance

See Also

`restoreCert` is this method’s inverse operation
parseCapabilities

*Construct a Node, using a passed in capabilities XML*

**Description**

Construct a Node, using a passed in capabilities XML

**Usage**

```r
parseCapabilities(x, ...)
```

```r
## S4 method for signature 'D1Node'
parseCapabilities(x, xml)
```

**Arguments**

- `x` The node to which capabilities should be applied.
- `...` (not yet used)
- `xml` The XML capabilities representing the node to be created

**Value**

The Node object with modified capabilities properties from the XML

---

**parseSolrResult**

*Parse Solr output into an R list*

**Description**

Solr output that is specified with a writer type of XML `&wt="xml"`.

**Usage**

```r
parseSolrResult(doc, ...)
```

```r
## S4 method for signature 'XMLInternalDocument'
parseSolrResult(doc, parse, ...)
```

**Arguments**

- `doc` The Solr result to parse, in XML format
- `...` (Not yet used.)
- `parse` A logical value, if TRUE the result is parsed to appropriate R types.

**Value**

`resultList` The Solr result as an R list
ping

Test if a node is online and accepting DataONE requests

Description

Test if a node is online and accepting DataONE requests

Usage

ping(x, ...)

## S4 method for signature 'D1Node'

ping(x)

Arguments

x The CNode or MNode to check
...
(Not yet used)

Value

logical A logical value set to TRUE if the node is up and FALSE if it is not

logical A logical value set to TRUE if the node is up and FALSE if it is not

Examples

## Not run:

cn <- CNode()

mn <- getNode(cn, "urn:node:KNB")

isAlive <- ping(mn)

## End(Not run)

query

Search DataONE for data and metadata objects

Description

The DataONE search index is searched for data that matches the specified query parameters.
Usage

query(x, ...)

## S4 method for signature 'D1Node'
query(
  x,
  solrQuery = as.character(NA),
  encode = TRUE,
  as = "list",
  parse = TRUE,
  searchTerms = as.character(NA),
  encodeReserved = FALSE,
  ...
)

Arguments

x The CNode or MNode instance to send the query to.
...
(Not yet used.)
solrQuery The query search terms, either as a string or as list with named members.
encode A logical, if TRUE then the query is URL encoded. The default is TRUE.
as The return type. Possible values: "json", "xml", "list" or "data.frame" with "list" as the default.
parse A logical value. If TRUE, then the result is parsed and converted to appropriate R data types. If FALSE, character values are returned.
searchTerms A list of name / value pairs (an alternative to solrQuery).
encodeReserved A logical, if TRUE then reserved characters in the query are URL encoded (FALSE is default). See URLencode for details.

Details

The "query" method sends a query to a DataONE search index that uses the Apache Solr search engine https://solr.apache.org/. This same Solr search engine is the underlying mechanism used by the DataONE online search tool available at https://search.dataone.org/.

The "solrQuery" argument is used to specify search terms that data of interest must match. This parameter uses Solr query terms, so some familiarity with Solr is helpful, however, fairly simple queries can be effective. This argument can be created as either a single character string containing the Solr query, for example: solrQuery = "q=id:doi*&rows=2&wt=json", or as a list of key value pairs: solrQuery = list(q = "id:doi*", rows = "2", wt = "json"). These two queries produce the same result.

As an alternative to specifying the Solr query terms using the "solrquery" argument, the "searchTerms" argument can be specified, which does not require any Solr syntax. This parameter is a list with query field / value pairs, i.e. searchTerms=list(abstract=kelp, attribute=biomass). The query fields can be listed for a DataONE node using getQueryEngineDescription. Either "searchTerms" or "solrQuery" must be specified.
The "as" argument is used to specify the query result to be returned as: "json", "xml", "list", "data.frame".

The "parsed" argument, if specified as TRUE, causes the query result to be converted to appropriate R data types. For example, if `as = "xml"` and `parsed = TRUE`, then the query result is returned as an R XMLInternalDocument, or if 'parsed = FALSE' then a character variable with the XML string is returned. Specify `as = "list"` to have the result parsed to an R list, with each list element containing one Solr query result of the total result set.

**Value**

search results as a list, data.frame or XML document

**Examples**

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

library(dataone)
cn <- CNode("FPROD")
queryParams <- list(q="id:doi*", rows="5",
   fq="(abstract:chlorophyll AND dateUploaded:[2000-01-01T00:00:00Z TO NOW])",
   fl="title,id,abstract,size,dateUploaded,attributeName")

# Return result as a list.
result <- query(cn, queryParams, as="list")

# Query and return the result as a data.frame of character values.
queryParams <- list(q="id:doi*", rows="3",
   fq="(abstract:chlorophyll AND dateUploaded:[2000-01-01T00:00:00Z TO NOW])",
   fl="title,id,abstract,size,dateUploaded,attributeName")
result <- query(cn, queryParams, as="data.frame", parse=FALSE)

# Return the result as JSON
queryParams <- "q=id:doi*&rows=2&wt=json"
result <- query(cn, queryParams, as="json")

# The following query shows how to embed quotes

queryParamList <- list(q="(attribute:lake) and (attribute:"Percent Nitrogen")", rows="1000",
   fl="title,id,abstract,size,dateUploaded,attributeName", wt="xml")
result <- query(cn, queryParamList, as="data.frame")

# The following query uses the searchTerms parameter

mn <- getNode(cn, "urn:node:KNB")
mySearchTerms <- list(abstract="kelp", attribute="biomass")
result <- query(mn, searchTerms=mySearchTerms, as="data.frame")

## End(Not run)
```
reserveIdentifier

Reserve a identifier that is unique in the DataONE network.

Description

The reserveIdentifier method contains the DataONE CN and reserves the specified identifier that the user has provided. Once a identifier has been reserved, it and can not be used by any other user.

Usage

reserveIdentifier(x, ...)

## S4 method for signature 'CNode'
reserveIdentifier(x, id)

## S4 method for signature 'D1Client'
reserveIdentifier(x, id)

Arguments

- **x** The coordinating node to query for its registered Member Nodes
- **...** Additional parameters.
- **id** The identifier that is to be reserved.

Details

This method requires a DataONE authentication token or X.509 Certificate. The reservation is made for the DataONE user identity that created the current authentication token or X.509 certificate.

Value

The reserved pid if it was successfully reserved, otherwise NULL

See Also

CNode class description.

Examples

```r
## Not run:
library(dataone)
library(uuid)\n-cn <- CNode("STAGING")
myId <- sprintf("urn:uuid:%s", UUIDgenerate())
newId <- reserveIdentifier(cn, myId)

## End(Not run)
```
## resolve

*Get a list of coordinating nodes holding a given pid.*

### Description

Returns a list of nodes (MNs or CNs) known to hold copies of the object identified by id.

### Usage

```r
resolve(x, ...)  
## S4 method for signature 'CNode'  
resolve(x, pid)
```

### Arguments

- **x**: a valid CNode object  
- **...**: Additional arguments (not yet used).  
- **pid**: the id of the identified object

### Value

A list of URLs that the object can be downloaded from, or NULL if the object is not found.

### Examples

```r
## Not run:  
library(dataone)  
library(dataone)  
cn <- CNode("STAGING")  
id <- "doi:10.6073/pasta/9a27a1615e8e4c366ad09febfba2fced"  
locations <- resolve(cn,id)  
## End(Not run)
```

## restoreAuth

*Restore authentication (after being disabled with obscureAuth).*

### Description

Restore authentication (after being disabled with obscureAuth).

### Usage

```r
restoreAuth(.Object)  
## S4 method for signature 'AuthenticationManager'  
restoreAuth(.Object)
```
Arguments

.object An AuthenticationManager instance

Value

The expiration date for the current authentication mechanism being used.

---

restoreCert

*Restore the CILogon client certificate by renaming it to its original location*

**Description**

Restores the x509 certificate that CILogon installs, which allows future interactions with nodes to be authenticated with the certificate. This function simply renames an obscured certificate file to its original location, allowing authenticated operations.

**Usage**

```r
restoreCert(x, ...)
```

## S4 method for signature 'CertificateManager'

```r
restoreCert(x)
```

Arguments

- `x` a CertificateManager instance
- `...` (Not yet used)

Value

the modified CertificateManager instance

**See Also**

*obscureCert* is this method’s inverse operation
setMNodeId

Set the member node identifier to be associated with the D1Client object.

Description

The member node identifier is the URN identifier used by DataONE to uniquely identifier a node, for example "urn:node:KNB" specifies the "Knowledge Network for Biodiversity" member node.

Usage

setMNodeId(x, id)

## S4 method for signature 'D1Client,character'
setMNodeId(x, id)

Arguments

x A D1Client object.

id A DataONE member node identifier.

Details

One Member Node can be associated with the client as the default to which data and metadata are written.

Author(s)

setMNodeId

See Also

D1Client class description.

setObsoletedBy

Set a pid as being obsoleted by another pid

Description

Updates the SystemMetadata 'obsoletedBy' property for an object, indicating that the object specified by pid has been obsoleted by the identifier in obsoletedByPid. CILogon https://cilogon.org/?skin=DataONE. See CertificateManager for details. In DataONE version 2.0, authentication tokens can also be used.
Usage

setObsoletedBy(x, pid, obsoletedByPid, ...)  

## S4 method for signature 'CNode, character'
setObsoletedBy(x, pid, obsoletedByPid, serialVersion)

Arguments

- **x** The CNode instance on which the object will be created
- **pid** The identifier of the object to be obsoleted
- **obsoletedByPid** The identifier of the object that obsoletes the object identified by pid.
- **...** (Not yet used)
- **serialVersion** The serial version of the system metadata of the pid being obsoleted.

Value

TRUE if the pid was obsoleted, otherwise FALSE is returned

See Also

CNode class description.

Description

This method should be called prior to creating the object in DataONE. When called before creating the object, adds a rule to the access policy that makes this object publicly readable. If called after creation, it will only change the system metadata locally, and will not have any effect on remotely uploaded copies of the D1Object.

Usage

## S4 method for signature 'D1Object'
setPublicAccess(x)

Arguments

- **x** D1Object

Value

D1Object with modified access rules
**showAuth**

*Display all authentication information*

**Description**

Display all authentication information

**Usage**

```r
showAuth(.Object, ...)
```

## S4 method for signature 'AuthenticationManager'

```r
showAuth(.Object, node)
```

**Arguments**

- `.Object`: An AuthenticationManager instance
- `...`: (Not yet used)
- `node`: A D1Node instance

---

**showClientSubject**

*Get DataONE Identity as Stored in the CILogon Certificate.*

**Description**

Returns Your Identity according to DataONE (and CILogon) as provided in the Subject field of the X.509 certificate. The value is a Distinguished Name, and can be used in all fields that require a user identity for access control authorization. If the certificate is missing or expired, then the subject 'public' is returned.

**Usage**

```r
showClientSubject(x, ...)
```

## S4 method for signature 'CertificateManager'

```r
showClientSubject(x)
```

**Arguments**

- `x`: a CertificateManager instance
- `...`: (Not yet used)
**updateObject**

Update an object on a Member Node, by creating a new object that replaces an original.

**Description**

This method provides the ability to update a data or metadata object to the Member Node provided in the 'x' parameter. In DataONE, both the original object and the new object are maintained, each with its own persistent identifier, and the 'obsoletes' field in the SystemMetadata is used to reflect the fact that the new object replaces the old. Both objects remain accessible.

**Usage**

```r
updateObject(x, ...) 
```

### S4 method for signature 'MNode'

```r
updateObject(x, pid, file = as.character(NA), newpid, sysmeta, dataobj = NULL)
```

**Arguments**

- `x` The MNode instance on which the object will be created
- `...` (Not yet used.)
- `pid` The identifier of the object to be updated
- `file` the absolute file location of the object to be uploaded
- `newpid` The identifier of the new object to be created
- `sysmeta` a SystemMetadata instance describing properties of the object
- `dataobj` a raw object to use for the upload, instead of the contents of the file argument.

**Details**

In the version 2.0 library and higher, this operation can utilize an 'dataone_token' option to provide credentials for write operations in DataONE. The authentication token is obtained from DataONE (see your profile on https://search.dataone.org). See the vignette("dataone-overview") for details. Alternatively, the version 1.0 approach of using an X.509 certificate in a default location of the file system can also be used. This certificate provides authentication credentials from CILogon [https://cilogon.org/?skin=DataONE](https://cilogon.org/?skin=DataONE). See vignette("dataone-overview") for details.

**Value**

A character containing the identifier if successful.
updateSystemMetadata

Note
Please see the vignette *upload-data* for an example: vignette("upload-data")

See Also
https://purl.dataone.org/architecture/apis/MN_APIs.html#MNStorage.update

updateSystemMetadata  Update the system metadata associated with an object.

Description
A modified SystemMetadata object can be sent to DataONE that contains updated information. This function allows updating of the system metadata without updating the object that it describes, so that mutable attributes such as accessPolicy can be updated easily.

Usage
updateSystemMetadata(x, ...)

## S4 method for signature 'MNode'
updateSystemMetadata(x, pid, sysmeta)

Arguments
x  The MNode instance from which the SystemMetadata will be downloaded
... (Not yet used.)
pid  The identifier of the object
sysmeta  a SystemMetadata instance with updated information.

Details
In the version 2.0 library and higher, this operation can utilize an ‘dataone_token’ option to provide credentials for write operations in DataONE. The authentication token is obtained from DataONE (see your profile on https://search.dataone.org). See the vignette("dataone-overview") for details. Alternatively, the version 1.0 approach of using an X.509 certificate in a default location of the file system can also be used. This certificate provides authentication credentials from CILogon https://cilogon.org/?skin=DataONE. See vignette("dataone-overview") for details.

Value
A logical value, TRUE if the operation was successful, FALSE if there was an error.

Note
Please see the vignette *upload-data* for an example: vignette("upload-data")
See Also

https://purl.dataone.org/architecture/apis/MN_APIs.html#MNStorage.updateSystemMetadata

uploadDataObject

Upload a DataObject to a DataONE member node.

Description

Upload a DataObject to a DataONE member node.

Usage

uploadDataObject(x, ...)

## S4 method for signature 'D1Client'
uploadDataObject(
  x,
  do,
  replicate = as.logical(FALSE),
  numberReplicas = NA,
  preferredNodes = NA,
  public = as.logical(FALSE),
  accessRules = NA,
  quiet = TRUE,
  ...
)

Arguments

x A D1Client instance.

... (Not yet used.)

do The DataObject instance to be uploaded to DataONE.

replicate A value of type "logical", if TRUE then DataONE will replicate this object to other member nodes

numberReplicas A value of type "numeric", for number of supported replicas.

preferredNodes A list of "character", each of which is the node identifier for a node to which a replica should be sent.

public A "logical" value - if TRUE then the uploaded object will be publicly readable.

accessRules Access rules of 'data.frame' that will be added to the access policy

quiet A 'logical'. If TRUE (the default) then informational messages will not be printed.
Value

id The id of the DataObject that was uploaded

See Also

D1Client class description.

Examples

```r
## Not run:
library(dataone)
library(datapack)
testdf <- data.frame(x=1:10, y=11:20)
csvfile <- tempfile(pattern = "file", tdir = tempdir(), fileext = ".csv")
write.csv(testdf, csvfile, row.names=FALSE)
d1c <- D1Client("STAGING", "urn:node:mmStageUCSB2")
do <- new("DataObject", format="text/csv", mnNodeId=getMNodeId(d1c), filename=csvfile)
# Upload a single DataObject to DataONE (requires authentication)
newId <- uploadDataObject(d1c, do, replicate=FALSE, preferredNodes=NA, public=TRUE)
## End(Not run)
```

uploadDataPackage

Upload a DataPackage to a DataONE member node.

Description

Upload all DataObjects contained in the DataPackage by calling uploadDataObject on each of the members. Also a resourceMap object is created from the recorded relationships between DataObjects, and this is uploaded as well.

Usage

```
uploadDataPackage(x, ...)
```

```r
# S4 method for signature 'D1Client'
uploadDataPackage(
x,
dp,
replicate = NA,
numberReplicas = NA,
preferredNodes = NA,
public = as.logical(FALSE),
accessRules = NA,
quiet = as.logical(TRUE),
resolveURI = as.character(NA),
packageId = as.character(NA),
as = "character",
```
Arguments

- `x`: A D1Client instance. (Not yet used.)
- `dp`: The DataPackage instance to be submitted to DataONE for creation.
- `replicate`: A value of type "logical", if TRUE then DataONE will replicate this object to other member nodes.
- `numberReplicas`: A value of type "numeric", for number of supported replicas.
- `preferredNodes`: A list of "character", each of which is the node identifier for a node to which a replica should be sent.
- `public`: A 'logical', if TRUE then all objects in this package will be accessible by any user.
- `accessRules`: Access rules of 'data.frame' that will be added to the access policy of each object in the datapackage.
- `quiet`: A 'logical'. If TRUE (the default) then informational messages will not be printed.
- `resolveURI`: A URI to prepend to identifiers (i.e. for use when creating the ResourceMap). See serializePackage
- `packageId`: A value of type "character" specifying a unique identifier to use for the uploaded package (resource map pid)
- `as`: A value of type "character" that specifies the return value. Possible values are "character" (the default) or "DataPackage".

Details

The DataPackage describes the collection of data object and their associated metadata object, with the relationships and members serialized into a document stored under, and retrievable with, the packageId as its own distinct object. Any objects in the data map that have a dateUploaded value are assumed to be pre-existing in the system, and skipped.

Value

- `id`: The identifier of the resource map for this data package

Note

Member objects are created serially, and most errors in creating one object will interrupt the create process for the whole, with the result that some members will be created, and the remainder not.

See Also

D1Client class description.
Examples

```r
## Not run:
library(dataone)
library(datapack)
dp <- new("DataPackage")
sampleData <- system.file("extdata/sample.csv", package="dataone")
dataObj <- new("DataObject", format="text/csv", file=sampleData)
dataObj <- setPublicAccess(dataObj)
sampleEML <- system.file("extdata/strix-pacific-northwest.xml", package="dataone")
metadataObj <- new("DataObject", format="eml://ecoinformatics.org/eml-2.1.1", file=sampleEML)
metadataObj <- setPublicAccess(metadataObj)
dp <- addMember(dp, do = dataObj, mo = metadataObj)
d1c <- D1Client("STAGING", "urn:node:mmStageUCSB2")
# Upload all members of the DataPackage to DataONE (requires authentication)
packageId <- uploadDataPackage(d1c, dp, replicate=TRUE, public=TRUE, numberReplicas=2)

## End(Not run)
```
Index

* classes
  CertificateManager-class, 15
  D1Object-class, 28

AbstractTableDescriber-class, 4
addData, DataPackage, D1Object-method, 5
archive, 6, 27
archive, D1Node-method (archive), 6
asDataFrame, 7, 28
asDataFrame, D1Object, AbstractTableDescriber-method (asDataFrame), 7
asDataFrame, D1Object, D1Object-method (asDataFrame), 7
auth_delete, 10
auth_get, 11
auth_head, 11
auth_post, 12
auth_put, 13
auth_put_post_delete, 13
AuthenticationManager, 8, 9, 37
AuthenticationManager, ANY-method (AuthenticationManager), 8
AuthenticationManager-class, 9

canRead, 28
canRead, D1Object-method, 14
CertificateManager, 6, 14, 16, 93
CertificateManager, ANY-method (CertificateManager), 14
CertificateManager-class, 15
CNode, 17, 17, 18, 24, 37, 61, 65, 70, 72, 76, 78, 80, 90, 94
CNode, ANY-method (CNode), 17
CNode, character-method (CNode), 17
CNode-class, 18
convert.csv, 19, 25
convert.csv, D1Client-method (convert.csv), 19
createD1Object, 20

createD1Object, D1Client, D1Object-method (createD1Object), 20
createDataPackage, 21, 25
createDataPackage, D1Client, DataPackage-method (createDataPackage), 21
createObject, 22, 83
createObject, MNode-method (createObject), 22
d1_errors, 30
D1Client, 19, 21, 23, 24–26, 29, 37, 42, 45, 46, 54, 55, 57, 59, 60, 64, 66, 79, 93, 99, 100
D1Client, ANY, ANY-method (D1Client), 23
D1Client, character, ANY-method (D1Client), 23
D1Client, character, character-method (D1Client), 23
D1Client, character, MNode-method (D1Client), 23
D1Client, CNode, MNode-method (D1Client), 23
D1Client-class, 24
D1Client-initialize (initialize, D1Client-method), 73
d1IdentifierSearch, 25
d1IdentifierSearch, D1Client-method (d1IdentifierSearch), 25
D1Node, 7, 26, 27, 37
D1Node, XMLInternalElementNode-method (D1Node), 26
D1Node-class, 27
D1Node-initialize (initialize, D1Node-method), 74
D1Object, 28, 28, 74
D1Object-class, 28
D1Object-initialize (initialize, D1Object-method), 74
getCert, 9, 51
getCert, AuthenticationManager-method (getCert), 51
ggetCertExpires, 16, 51
ggetCertExpires, CertificateManager-method (getCertExpires), 51
ggetCertInfo, 10, 52
ggetCertInfo, AuthenticationManager-method (getCertInfo), 52
ggetCertLocation, 16, 40, 52
ggetCertLocation, CertificateManager-method (getCertLocation), 52
ggetChecksum, 18, 27, 53
ggetChecksum, CNode-method (getChecksum), 53
ggetChecksum, MNode-method (getChecksum), 53
ggetCN, 54
ggetCN, D1Client-method (getCN), 54
ggetD1Object, 55
ggetD1Object, D1Client-method (getD1Object), 55
ggetData, 28
ggetData, D1Object-method, 56
ggetDataObject, 25, 56
ggetDataObject, D1Client-method (getDataObject), 56
ggetDataPackage, 25, 58
ggetDataPackage, D1Client-method (getDataPackage), 58
ggetEndpoint, 25, 59
ggetEndpoint, D1Client-method (getEndpoint), 59
ggetErrorDescription, 60
ggetFormat, 18, 61
ggetFormat, CNode-method (getFormat), 61
ggetFormatId, 28
ggetFormatId, D1Object-method, 62
ggetIdentifer, 28
ggetIdentifer, D1Object-method, 62
ggetMetadataMember, 25, 63
ggetMetadataMember, D1Client, DataPackage-method (getMetadataMember), 63
ggetMNode, 63
ggetMNode, D1Client, ANY-method (getMNode), 63
ggetMNode, D1Client, character-method (getMNode), 63
ggetMNode, 18, 64
ggetMNodeId, 25, 64, 65
ggetMNodeId, D1Client-method (getMNodeId), 65
ggetObject, 18, 27, 66, 83
ggetObject, CNode-method (getObject), 66
ggetObject, MNode-method (getObject), 66
ggetPackage, 67, 83
ggetPackage, MNode-method (getPackage), 67
ggetQueryEngineDescription, 27, 68, 88
ggetQueryEngineDescription, D1Node-method (getQueryEngineDescription), 68
ggetSystemMetadata, 18, 27, 69
ggetSystemMetadata, CNode-method (getSystemMetadata), 69
ggetSystemMetadata, MNode-method (getSystemMetadata), 69
ggetToken, 9, 70
ggetToken, AuthenticationManager-method (getToken), 70
ggetTokenInfo, 10, 71
ggetTokenInfo, AuthenticationManager-method (getTokenInfo), 71

hasReservation, 18, 72
hasReservation, CNode-method (hasReservation), 72

initialize, D1Client-method, 73
initialize, D1Node-method, 74
isAuthExpired, 9, 75
isAuthExpired, AuthenticationManager-method (isAuthExpired), 75
isAuthorized, 18, 27, 75
isAuthorized, D1Node-method (isAuthorized), 75
isAuthValid, 9, 76
isAuthValid, AuthenticationManager-method (isAuthValid), 76
isCertExpired, 16, 77
isCertExpired, CertificateManager-method (isCertExpired), 77

listFormats, 18, 78, 78
listFormats, CNode-method (listFormats), 78
listMemberNodes, 25, 78
INDEX

105

listMemberNodes, D1Client-method (listMemberNodes), 78
listObjects, 78, 80
listQueryEngines, 78, 81
listQueryEngines, D1Node-method (listQueryEngines), 81
listNodes, 78, 79
listNodes, CNode-method (listNodes), 79
listObjects, 78, 80
listObjects, D1Node-method (listObjects), 80
listNodes, 79
listNodes, CNode-method (listNodes), 79
listObjects, 78, 80
listObjects, D1Node-method (listObjects), 80
listQueryEngines, 78, 81
listQueryEngines, D1Node-method (listQueryEngines), 81

MNode, 24, 37, 68, 82, 83
MNode, character-method (MNode), 82
MNode, D1Node-method (MNode), 82
MNode-class, 83

obscureAuth, 9, 84
obscureAuth, AuthenticationManager-method (obscureAuth), 84
obscureCert, 16, 85, 92
obscureCert, CertificateManager-method (obscureCert), 85

parseCapabilities, 86
parseCapabilities, D1Node-method (parseCapabilities), 86
parseSolrResult, 86
parseSolrResult, XMLInternalDocument-method (parseSolrResult), 86
ping, 27, 87
ping, D1Node-method (ping), 87

query, 27, 87
query, D1Node-method (query), 87

reserveIdentifier, 18, 25, 90
reserveIdentifier, CNode-method (reserveIdentifier), 90
reserveIdentifier, D1Client-method (reserveIdentifier), 90
resolve, 18, 91
resolve, CNode-method (resolve), 91
restoreAuth, 10, 91
restoreAuth, AuthenticationManager-method (restoreAuth), 91
restoreCert, 16, 85, 92
restoreCert, CertificateManager-method (restoreCert), 92

updateObject, 83, 96
updateObject, D1Node-method (updateObject), 96
updateSystemMetadata, 83, 97
updateSystemMetadata, MNode-method (updateSystemMetadata), 97
uploadDataObject, 25, 98, 99
uploadDataObject, D1Client-method (uploadDataObject), 98
uploadDataPackage, 24, 25, 99
uploadDataPackage, D1Client-method (uploadDataPackage), 99

setMNodeId, 93
setMNodeId, D1Client, character-method (setMNodeId), 93
setObsoletedBy, 18, 93
setObsoletedBy, CNode, character-method (setObsoletedBy), 93
setPublicAccess, 28
setPublicAccess, D1Object-method, 94
showAuth, 10, 95
showAuth, AuthenticationManager-method (showAuth), 95
showClientSubject, 16, 95
showClientSubject, CertificateManager-method (showClientSubject), 95

updateObject, 83, 96
updateObject, MNode-method (updateObject), 96
updateSystemMetadata, 83, 97
updateSystemMetadata, MNode-method (updateSystemMetadata), 97
uploadDataObject, 25, 98, 99
uploadDataObject, D1Client-method (uploadDataObject), 98
uploadDataPackage, 24, 25, 99
uploadDataPackage, D1Client-method (uploadDataPackage), 99

serializePackage, 100