Package ‘EDIutils’

October 10, 2023

Title An API Client for the Environmental Data Initiative Repository

Version 1.0.3

Description A client for the Environmental Data Initiative repository REST API. The 'EDI' data repository <https://portal.edirepository.org/nis/home.jsp> is for publication and reuse of ecological data with emphasis on metadata accuracy and completeness. It is built upon the 'PASTA+' software stack <https://pastaplus-core.readthedocs.io/en/latest/index.html#> and was developed in collaboration with the US 'LTER' Network <https://lternet.edu/>. 'EDIutils' includes functions to search and access existing data, evaluate and upload new data, and assist other data management tasks common to repository users.

Imports curl, httr, jsonlite, xml2

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.2.3

Suggests knitr, readr, vcr, rmarkdown, roxygen2, testthat

URL https://github.com/ropensci/EDIutils,
https://docs.ropensci.org/EDIutils/

BugReports https://github.com/ropensci/EDIutils/issues

VignetteBuilder knitr

Language en-US

NeedsCompilation no

Author Colin Smith [aut, cre] (<https://orcid.org/0000-0003-2261-9931>),
Corinna Gries [ctb] (<https://orcid.org/0000-0002-9091-6543>),
Jasmine Lai [rev] (<https://orcid.org/0000-0001-8888-547X>),
Rodrigo Pires [rev] (<https://orcid.org/0000-0001-7384-6849>)

Maintainer Colin Smith <colin.smith@wisc.edu>

Repository CRAN

Date/Publication 2023-10-10 18:00:02 UTC
R topics documented:

- check_status_create ........................................ 3
- check_status_evaluate ........................................ 5
- check_status_update .......................................... 6
- create_data_package .......................................... 7
- create_data_package_archive ................................. 9
- create_dn ...................................................... 9
- create_event_subscription ................................... 10
- create_journal_citation .................................... 11
- create_reservation .......................................... 13
- delete_event_subscription .................................. 14
- delete_journal_citation .................................... 15
- delete_reservation ......................................... 16
- evaluate_data_package ..................................... 17
- execute_event_subscription ................................. 19
- get_audit_count ............................................ 21
- get_audit_record ............................................ 22
- get_audit_report ............................................ 23
- get_docid_reads ............................................. 25
- get_event_subscription ..................................... 26
- get_event_subscription_schema ............................ 27
- get_journal_citation ....................................... 28
- get_packageid_reads ....................................... 29
- get_provenance_metadata ................................... 30
- get_recent_uploads .......................................... 31
- is_authorized ............................................... 32
- list_active_reservations ................................... 33
- list_data_descendants ...................................... 34
- list_data_entities .......................................... 35
- list_data_package_citations ............................... 36
- list_data_package_identifiers ............................. 37
- list_data_package_revisions .............................. 38
- list_data_package_scopes .................................. 39
- list_data_sources .......................................... 40
- list_deleted_data_packages ............................... 41
- list_principal_owner_citations ......................... 42
- list_recent_changes ........................................ 43
- list_recent_uploads ........................................ 44
- list_reservation_identifiers .............................. 45
- list_service_methods ....................................... 46
- list_user_data_packages .................................... 47
- list_working_on ............................................ 48
- login ......................................................... 49
- logout ......................................................... 50
- query_event_subscriptions ................................ 51
- read_data_entity ........................................... 52
- read_data_entity_checksum ................................ 54
check_status_create

Check data package creation status

Description

Check data package creation status

Usage

check_status_create(transaction, wait = TRUE, env = "production")

Arguments

transaction (character) Transaction identifier
wait (logical) Wait for evaluation to complete? See details below.
env (character) Repository environment. Can be: "production", "staging", or "development".
check_status_create

Details

If `wait = TRUE`, then the function will enter a "while" loop checking every 2 seconds for the completed evaluation report. If `wait = FALSE`, then the function will only check once and return the result.

Value

(logical) TRUE if creation has completed, FALSE if in progress, and error if an error was encountered while processing the request

Note

User authentication is required (see `login()`)

See Also

Other Evaluation and Upload: `check_status_evaluate()`, `check_status_update()`, `create_data_package()`, `evaluate_data_package()`, `update_data_package()`

Examples

```{r}
## Not run:

login()

# Create data package
transaction <- create_data_package(
    eml = paste0(tempdir(), "/edi.595.1.xml"),
    env = "staging"
)
transaction
#> [1] "create_1639666765080210573__edi.595.1"

# Check creation status
status <- check_status_create(
    transaction = transaction,
    env = "staging"
)
status
#> [1] TRUE

logout()

## End(Not run)
```
Description
Check status of data package evaluation

Usage
check_status_evaluate(transaction, wait = TRUE, env = "production")

Arguments
- transaction (character) Transaction identifier
- wait (logical) Wait for evaluation to complete? See details below.
- env (character) Repository environment. Can be: "production", "staging", or "development".

Details
If wait = TRUE, then the function will enter a "while" loop checking every 2 seconds for the completed evaluation report. If wait = FALSE, then the function will only check once and return the result.

Value
(logical) TRUE if evaluation has completed, FALSE if in progress, and error if an error was encountered while processing the request

Note
User authentication is required (see login())

See Also
Other Evaluation and Upload: check_status_create(), check_status_update(), create_data_package(), evaluate_data_package(), update_data_package()

Examples
## Not run:
login()

# Evaluate data package
transaction <- evaluate_data_package(
  eml = paste0(tempdir(), "/edi.595.1.xml"),
  env = "staging"
check_status_update

Check data package update status

Description
Check data package update status

Usage
check_status_update(transaction, wait = TRUE, env = "production")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>(character) Transaction identifier</td>
</tr>
<tr>
<td>wait</td>
<td>(logical) Wait for evaluation to complete? See details below.</td>
</tr>
<tr>
<td>env</td>
<td>(character) Repository environment. Can be: &quot;production&quot;, &quot;staging&quot;, or &quot;development&quot;.</td>
</tr>
</tbody>
</table>

Details
If wait = TRUE, then the function will enter a "while" loop checking every 2 seconds for the completed evaluation report. If wait = FALSE, then the function will only check once and return the result.

Value

(logical) TRUE if the update has completed, FALSE if in progress, and error if an error was encountered while processing the request

Note
User authentication is required (see login())
create_data_package

See Also
Other Evaluation and Upload: \texttt{check\_status\_create()}, \texttt{check\_status\_evaluate()}, \texttt{create\_data\_package()}, \texttt{evaluate\_data\_package()}, \texttt{update\_data\_package()}

Examples

```r
## Not run:

login()

# Update data package
transaction <- update_data_package(
  eml = paste0(tempdir(), "/edi.595.2.xml"),
  env = "staging"
)
transaction
#> [1] "update_edi.595_163966788658131920__edi.595.2"

# Check update status
status <- check_status_update(
  transaction = transaction,
  env = "staging"
)
status
#> [1] TRUE

logout()

## End(Not run)
```

create_data_package  Create data package

Description
Create data package

Usage

```r
create_data_package(eml, env = "production")
```

Arguments

\begin{itemize}
\item \texttt{eml}  (character) Full path to an EML file describing the data package to be created
\item \texttt{env}  (character) Repository environment. Can be: "production", "staging", or "development".
\end{itemize}
Details

Each data entity described in eml must be accompanied by a web accessible URL at the EML XPath ".//physical/distribution/online/url". The EDI data repository downloads the data entities via this URL. The URLs must be static and not have any redirects otherwise the data entities will not be downloaded.

Value

transaction (character) Transaction identifier. May be used in a subsequent call to check_status_create() to determine the operation status.

Note

User authentication is required (see login())

See Also

Other Evaluation and Upload: check_status_create(), check_status_evaluate(), check_status_update(), evaluate_data_package(), update_data_package()

Examples

```r
## Not run:

login()

# Create data package
transaction <- create_data_package(
  eml = paste0(tempdir(), "/edi.595.1.xml"),
  env = "staging"
)
transaction
#> [1] "create_163966765080210573__edi.595.1"

# Check creation status
status <- check_status_create(
  transaction = transaction,
  env = "staging"
)
status
#> [1] TRUE

logout()

## End(Not run)
```
create_data_package_archive

Create data package archive (zip)

Description
This function is DEPRECATED.

Usage
create_data_package_archive(packageId, env = "production")

Arguments
packageId (character) Data package identifier
env (character) Repository environment. Can be: "production", "staging", or "development".

Value
transaction (character) Transaction identifier.

See Also
Other Miscellaneous: create_dn(), is_authorized()

create_dn

Create a users distinguished name

Description
Create a users distinguished name

Usage
create_dn(userId, ou = "EDI")

Arguments
userId (character) User identifier of an EDI data repository account
ou (character) Organizational unit in which userId belongs. Can be "EDI" or "LTER". All userId issued after "2020-05-01" have ou = "EDI".

Value
(character) Distinguished name
create_event_subscription

Description
Create event subscription

Usage
create_event_subscription(packageId, url, env = "production")

Arguments
- packageId (character) Data package identifier
- url (character) Where the event notification will be sent
- env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(numeric) Event subscription identifier

Note
User authentication is required (see login())
The url must have "http" as its scheme and must be able to receive POST requests with MIME type text/plain. Additionally, because the url will be passed in an XML body, some characters must be escaped, such as ampersands from & to &.

See Also
Other Event Notifications: delete_event_subscription(), execute_event_subscription(), get_event_subscription_schema(), get_event_subscription(), query_event_subscriptions()
create_journal_citation

Examples

```r
## Not run:

login()

# Create subscription
subscriptionId <- create_event_subscription(
  packageId = "knb-lter-vcr.340.1",
  url = "https://my.webserver.org/",
  env = "staging"
)
subscriptionId
#> [1] 48

# Execute subscription
execute_event_subscription(subscriptionId, env = "staging")
#> [1] TRUE

# Delete subscription
delete_event_subscription(subscriptionId, env = "staging")
#> [1] TRUE

logout()

## End(Not run)
```

----

create_journal_citation

Create journal citation

Description

Create journal citation

Usage

```r
create_journal_citation(
  packageId,
  articleDoi = NULL,
  articleUrl = NULL,
  articleTitle = NULL,
  journalTitle = NULL,
  relationType,
  env = "production"
)
```
Arguments

- **packageId**: (character) Data package identifier
- **articleDoi**: (character) Article DOI. Required if `articleUrl` is missing.
- **articleUrl**: (character) Article URL. Required if `articleDoi` is missing.
- **articleTitle**: (character) Article title
- **journalTitle**: (character) Journal title
- **relationType**: (character) Relation between citation and data package. Can be: "IsCitedBy" - this data package is formally cited in the manuscript; "IsDescribedBy" - this data package is explicitly described within the manuscript; "IsReferencedBy" - this data package is implicitly described within the manuscript.
- **env**: (character) Repository environment. Can be: "production", "staging", or "development".

Details

Creates a new journal citation entry in the EDI data repository

Value

(numeric) Journal citation identifier

Note

User authentication is required (see `login()`)

See Also

Other Journal Citations: `delete_journal_citation()`, `get_journal_citation()`, `list_data_package_citations()`, `list_principal_owner_citations()`

Examples

```r
## Not run:

login()

# Create journal citation
journalCitationId <- create_journal_citation(
  packageId = "edi.17.1",
  articleDoi = "10.1890/11-1026.1",
  articleTitle = "Corridors promote fire via connectivity and edge effects",
  journalTitle = "Ecological Applications",
  relationType = "IsCitedBy",
  env = "staging"
)
journalCitationId
#> [1] 74

# Delete journal citation
```
create_reservation

```r
delete_journal_citation(journalCitationId, env = "staging")
## End(Not run)
```

---

**create_reservation**

---

**Description**

Reserves the next available identifier for the specified scope

**Usage**

```r
create_reservation(scope, env = "production")
```

**Arguments**

- **scope** (character) Scope of data package
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

**Value**

(numeric) Identifier of reserved data package

**Note**

User authentication is required (see `login()`)

**See Also**

Other Identifier Reservations: `delete_reservation()`, `list_active_reservations()`, `list_reservation_identifiers()`

**Examples**

```r
## Not run:

login()

# Create reservation
identifier <- create_reservation(scope = "edi", env = "staging")
identifier
## End(Not run)
```

```r
logout()

## End(Not run)
```
delete_reservation(scope = "edi", identifier = identifier, env = "staging")
#> [1] 604
logout()
## End(Not run)

---

**delete_event_subscription**

*Delete event subscription*

### Description
Delete event subscription

### Usage

```r
delete_event_subscription(subscriptionId, env = "production")
```

### Arguments

- `subscriptionId` (numeric) Event subscription identifier
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

### Details
After "deletion", the subscription might still exist in the subscription database, but it will be inactive - it will not conflict with future creation requests, it cannot be read, and it will not be notified of events.

### Value

(logical) TRUE if the event subscription was deleted

### Note
User authentication is required (see `login()`)

### See Also

Other Event Notifications: `create_event_subscription()`, `execute_event_subscription()`, `get_event_subscription_schema()`, `get_event_subscription()`, `query_event_subscriptions()`
Examples

```r
## Not run:

login()

# Create subscription
subscriptionId <- create_event_subscription(
  packageId = "knb-lter-vcr.340.1",
  url = "https://my.webserver.org/",
  env = "staging"
)
subscriptionId
#> [1] 48

# Execute subscription
execute_event_subscription(
  subscriptionId = subscriptionId,
  env = "staging"
)
#> [1] TRUE

# Delete subscription
delete_event_subscription(subscriptionId, env = "staging")
#> [1] TRUE

logout()

## End(Not run)
```

---

`delete_journal_citation`  
*Delete journal citation*

**Description**

Delete journal citation

**Usage**

```r
delete_journal_citation(journalCitationId, env = "production")
```

**Arguments**

- `journalCitationId`  
  (numeric) Journal citation identifier
- `env`  
  (character) Repository environment. Can be: "production", "staging", or "development".
**delete_reservation**

**Value**

(logical) TRUE if deleted

**Note**

User authentication is required (see `login()`)

**See Also**

Other Journal Citations: `create_journal_citation()`, `get_journal_citation()`, `list_data_package_citations()`, `list_principal_owner_citations()`

**Examples**

```r
## Not run:

login()

# Create journal citation
journalCitationId <- create_journal_citation(
  packageId = "edi.17.1",
  articleDoi = "https://doi.org/10.1890/11-1026.1",
  articleTitle = "Corridors promote fire via connectivity and edge effects",
  journalTitle = "Ecological Applications",
  relationType = "IsCitedBy",
  env = "staging"
)
journalCitationId
#> [1] 74

# Delete journal citation
delete_journal_citation(journalCitationId, env = "staging")
#> [1] TRUE

logout()

## End(Not run)
```

---

**delete_reservation**

Delete reservation

**Description**

Delete reservation

**Usage**

`delete_reservation(scope, identifier, env = "production")`
evaluate_data_package

Arguments

scope (character) Scope of data package
identifier (numeric) Identifier of data package
env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(numeric) The deleted reservation identifier value

Note

User authentication is required (see `login()`). The same user who originally authenticated to create the reservation must authenticate to delete it.

See Also

Other Identifier Reservations: `create_reservation()`, `list_active_reservations()`, `list_reservation_identifiers()`

Examples

```r
## Not run:

login()

# Create reservation
defifer <- create_reservation(scope = "edi", env = "staging")
identifier
#> [1] 604

# Delete reservation
delete_reservation(scope = "edi", identifier = identifier, env = "staging")
#> [1] 604

logout()

## End(Not run)
```

---

evaluate_data_package  Evaluate data package

Description

Evaluate data package

Usage

evaluate_data_package(eml, useChecksum = FALSE, env = "production")
evaluate_data_package

Arguments

- **eml** (character) Full path to an EML file describing the data package to be evaluated
- **useChecksum** (logical) Use data entities from a previous version of the data package? See details below.
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

Details

Each data entity described in **eml** must be accompanied by a web accessible URL at the EML XPath ".//physical/distribution/online/url". The EDI data repository downloads the data entities via this URL. The URLs must be static and not have any redirects otherwise the data entities will not be downloaded.

An optional query parameter, "useChecksum", can be appended to the URL. When specified, the useChecksum query parameter directs the repository to determine whether it can use an existing copy of a data entity from a previous revision of the data package based on matching a metadata-documented checksum value (MD5 or SHA-1) to the checksum of the existing copy. If a match is found, the repository will skip the upload of the data entity from the remote URL and instead use its matching copy. Specifying "useChecksum" can save time by eliminating data uploads, but clients should take care to ensure that metadata-documented checksum values are accurate and up to date.

Value

- **transaction** (character) Transaction identifier. May be used in a subsequent call to:
  - `check_status_evaluate()` to determine the operation status
  - `read_evaluate_report()` to read the evaluation report
  - `read_evaluate_report_summary()` to summarize the evaluation report and raise exceptions

Note

User authentication is required (see `login()`)

See Also

Other Evaluation and Upload: `check_status_create()`, `check_status_evaluate()`, `check_status_update()`, `create_data_package()`, `update_data_package()`

Examples

```r
## Not run:

login()

transaction <- evaluate_data_package(
  eml = paste0(tempdir(), "/edi.595.1.xml"),
  env = "staging"
)
transaction
```
execute_event_subscription

#> [1] "evaluate_163966785813042760"

# Check evaluation status
status <- check_status_evaluate(transaction, env = "staging")
status
#> [1] TRUE

# Read evaluation report
report <- read_evaluate_report(transaction, env = "staging")
report
#> <xml_document>
#> <qualityReport schemaLocation="eml://ecoinformatics.org/qualityReport ...
#> [1] <creationDate>2021-12-15T17:46:33</creationDate>
#> [2] <packageId>edi.595.1</packageId>
#> [3] <includeSystem>lter</includeSystem>
#> [4] <includeSystem>knb</includeSystem>
#> <qualityCheck qualityType="metadata" system=" ...
#> [6] <entityReport>
#> <entityName>data.txt</entityName>
#> <qualityC ...

# Summarize evaluation report
read_evaluate_report_summary(transaction, env = "staging")
#> ===================================================
#> EVALUATION REPORT
#> ===================================================
#> PackageId: edi.595.1
#> Report Date/Time: 2021-12-15T17:46:33
#> Total Quality Checks: 29
#> Valid: 21
#> Info: 8
#> Warn: 0
#> Error: 0

logout()

## End(Not run)

execute_event_subscription

Execute event subscription

Description

Execute event subscription

Usage

execute_event_subscription(subscriptionId, env = "production")
execute_event_subscription

Arguments

subscriptionId (numeric) Event subscription identifier
env (character) Repository environment. Can be: "production", "staging", or "development".

Details

Upon notification, the event manager queries its database for the subscription matching the specified subscriptionId. POST requests are then made (asynchronously) to the matching subscription.

Value

(logical) TRUE if the event subscription was executed

Note

User authentication is required (see login())

See Also

Other Event Notifications: create_event_subscription(), delete_event_subscription(),
get_event_subscription_schema(), get_event_subscription(), query_event_subscriptions()

Examples

## Not run:

login()

# Create subscription
subscriptionId <- create_event_subscription(
  packageId = "knb-lter-vcr.340.1",
  url = "https://my.webserver.org/",
  env = "staging"
)
subscriptionId
#> [1] 48

# Execute subscription
execute_event_subscription(
  subscriptionId = subscriptionId,
  env = "staging"
)
#> [1] TRUE

# Delete subscription
delete_event_subscription(subscriptionId, env = "staging")
#> [1] TRUE

logout()}
get_audit_count

## End(Not run)

---

**get_audit_count**

### Get audit count

**Description**

Get audit count

**Usage**

`get_audit_count(query, env = "production")`

**Arguments**

- **query** (character) Query (see details below)
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

**Details**

Query parameters are specified as key=value pairs, multiple pairs must be delimited with ampersands (&), and only a single value should be specified for a particular key. The following query parameter keys are allowed:

- **category** - Can be: debug, info, error, warn
- **service** - Any of the EDI data repository services
- **serviceMethod** - Any of the EDI data repository service Resource class JAX-RS methods
- **user** - Any user
- **group** - Any group
- **authSystem** - A valid auth system identifier
- **status** - A valid HTTP Response Code
- **resourceId** - An EDI data repository resource identifier, e.g. https://pasta.lternet.edu/package/eml/knb-lter-and/2719/6, or a substring thereof (see details below)
- **fromTime** - An ISO8601 timestamp
- **toTime** - An ISO8601 timestamp
- **limit** - A positive whole number

The query parameters fromTime and optionally toTime should be used to indicate a time span. When toTime is absent, the count will include of all matching records up to the current time. Either of these parameters may only be used once. The query parameter limit sets an upper limit on the number of audit records returned. For example, "limit=1000". The query parameter resourceId will match any audit log entry whose resourceId value contains the specified string value. Thus, a query parameter of "resourceId=knb-lter-and" will match any audit log entry whose resourceId value contains the substring "knb-lter-and", while a query parameter of "resourceId=knb-lter-and/2719/6" will match any audit log entry whose resourceId value contains the substring "knb-lter-and/2719/6".
Value

(numeric) Returns a count of the number of audit records matching the query parameters as specified in the request.

Note

User authentication is required (see `login()`)

See Also

Other Audit Manager Services: `get_audit_record()`, `get_audit_report()`, `get_docid_reads()`, `get_packageid_reads()`, `get_recent_uploads()`

Examples

```r
## Not run:

login()

# Count the number of warnings issued between 2021-12-01 and 2021-12-05
res <- get_audit_count(
  query = "category=warn&fromTime=2021-12-01&toTime=2021-12-05"
)
res
#> [1] 10022

logout()

## End(Not run)
```

---

**get_audit_record**

*Get audit record*

**Description**

Get audit record

**Usage**

```r
get_audit_record(oid, as = "data.frame", env = "production")
```

**Arguments**

- **oid** (numeric) Audit identifier
- **as** (character) Format of the returned object. Can be: "data.frame" or "xml".
- **env** (character) Repository environment. Can be: "production", "staging", or "development".
get_audit_report

Value

(data.frame or xml_document) An audit record

Note

User authentication is required (see login())

See Also

Other Audit Manager Services: get_audit_count(), get_audit_report(), get_docid_reads(), get_packageid_reads(), get_recent_uploads()

Examples

## Not run:

login()

# Get audit report
auditReport <- get_audit_report(oid = "121606334")

logout()

## End(Not run)

---

get_audit_report Get audit report

Description

Get audit report

Usage

get_audit_report(query, as = "data.frame", env = "production")

Arguments

query (character) Query (see details below)
as (character) Format of the returned object. Can be: "data.frame" or "xml".
env (character) Repository environment. Can be: "production", "staging", or "development".
Details

Query parameters are specified as key=value pairs, multiple pairs must be delimited with ampersands (&), and only a single value should be specified for a particular key. The following query parameter keys are allowed:

- **category** - Can be: debug, info, error, warn
- **service** - Any of the EDI data repository services
- **serviceMethod** - Any of the EDI data repository service Resource class JAX-RS methods
- **user** - Any user
- **group** - Any group
- **authSystem** - A valid auth system identifier
- **status** - A valid HTTP Response Code
- **resourceId** - An EDI data repository resource identifier, e.g. https://pasta.lternet.edu/package/eml/knb-lter-and/2719/6, or a thereof (see details below)
- **fromTime** - An ISO8601 timestamp
- **toTime** - An ISO8601 timestamp
- **limit** - A positive whole number

The query parameters fromTime and optionally toTime should be used to indicate a time span. When toTime is absent, the report will consist of all matching records up to the current time. Either of these parameters may only be used once. The query parameter limit sets an upper limit on the number of audit records returned. For example, "limit=1000". The query parameter resourceId will match any audit log entry whose resourceId value contains the specified string value. Thus, a query parameter of "resourceId=knb-lter-and" will match any audit log entry whose resourceId value contains the substring "knb-lter-and", while a query parameter of "resourceId=knb-lter-and/2719/6" will match any audit log entry whose resourceId value contains the substring "knb-lter-and/2719/6".

Value

(data.frame or xml_document) Zero or more audit records matching the query parameters as specified in the request (see details below).

Note

User authentication is required (see login())

See Also

Other Audit Manager Services: get_audit_count(), get_audit_record(), get_docid_reads(), get_packageid_reads(), get_recent Uploads()
### Examples

```r
## Not run:

```login()

``` # Get audit report for data reads between 2021-12-01 and 2021-12-02
query <- "serviceMethod=readDataEntity&fromTime=2021-12-01&toTime=2021-12-02"
auditReport <- get_audit_report(query)

```logout()

``` # End(Not run)

---

<table>
<thead>
<tr>
<th>get_docid_reads</th>
<th>Get doc ID reads</th>
</tr>
</thead>
</table>

#### Description

Get doc ID reads

#### Usage

```r
get_docid_reads(scope, identifier, as = "data.frame", env = "production")
```

#### Arguments

- **scope**: (character) Scope of data package
- **identifier**: (numeric) Identifier of data package
- **as**: (character) Format of the returned object. Can be: "data.frame" or "xml".
- **env**: (character) Repository environment. Can be: "production", "staging", or "development".

#### Value

(data.frame or xml_document) Summary of all the successful reads (total reads and non-robot reads) for all the resources of a given scope and identifier.

#### See Also

Other Audit Manager Services: `get_audit_count()`, `get_audit_record()`, `get_audit_report()`, `get_packageid_reads()`, `get_recent_uploads()`
get_event_subscription

Get event subscription

Description

Get event subscription

Usage

get_event_subscription(subscriptionId, as = "data.frame", env = "production")

Arguments

subscriptionId (numeric) Event subscription identifier
as (character) Format of the returned object. Can be: "data.frame" or "xml".
env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Subscription metadata

Note

User authentication is required (see login())

See Also

Other Event Notifications: create_event_subscription(), delete_event_subscription(), execute_event_subscription(), get_event_subscription_schema(), query_event_subscriptions()
Examples

## Not run:

```r
login()

# Get subscription
subscription <- get_event_subscription(
  subscriptionId = 21,
  env = "staging"
)

logout()

## End(Not run)
```

---

**get_event_subscription_schema**

*Get event subscription schema*

Description

Get event subscription schema

Usage

```r
get_event_subscription_schema(env = "production")
```

Arguments

- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(xml_document) Schema for event subscription creation request entities.

See the `xml2` library for more on working with XML.

See Also

Other Event Notifications: `create_event_subscription()`, `delete_event_subscription()`, `execute_event_subscription()`, `get_event_subscription()`, `query_event_subscriptions()`
Examples

```r
## Not run:

# Get schema
schema <- get_event_subscription_schema()
schema
#> {xml_document}
#> <schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
#> [1] <xs:element name="subscription"/>
#> <xs:complexType>
#> <xs: ...  

# Show schema structure
xml2::xml_structure(schema)
#> <schema xmlns:xs>
#> <element [name]
#> <complexType>
#> <all>
#> <element [name, type, minOccurs, maxOccurs]>
#> <element [name, type, minOccurs, maxOccurs]>
#> <attribute [name, type, use, fixed]>

## End(Not run)
```

---

**get_journal_citation**  
*Get journal citation*

**Description**

Get journal citation

**Usage**

```r
get_journal_citation(journalCitationId, as = "data.frame", env = "production")
```

**Arguments**

- `journalCitationId`  
  (numeric) Journal citation identifier

- `as`  
  (character) Format of the returned object. Can be: "data.frame" or "xml".

- `env`  
  (character) Repository environment. Can be: "production", "staging", or "development".

**Value**

(data.frame or xml_document) Journal citation

**See Also**

Other Journal Citations: `create_journal_citation()`, `delete_journal_citation()`, `list_data_package_citations()`, `list_principal_owner_citations()`
get_packageid_reads

Examples

## Not run:

# Get citation
journalCitation <- get_journal_citation(381)

## End(Not run)

get_packageid_reads  Get package ID reads

Description

Get package ID reads

Usage

get_packageid_reads(packageId, as = "data.frame", env = "production")

Arguments

packageId  (character) Data package identifier
as  (character) Format of the returned object. Can be: "data.frame" or "xml".
env  (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Summary of all the successful reads (total reads and non-robot reads) of packageId

See Also

Other Audit Manager Services: get_audit_count(), get_audit_record(), get_audit_report(), get_docid_reads(), get_recent_uploads()

Examples

## Not run:

# Get packageId reads
resourceReads <- get_packageid_reads("knb-lter-sgs.817.17")

## End(Not run)
get_provenance_metadata

Get provenance metadata

Description

Generates the provenance metadata of a source data package

Usage

get_provenance_metadata(packageId, env = "production")

Arguments

packageId (character) Data package identifier
env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(xml_document) Provenance metadata of packageId, representing a <methodStep> element that can be inserted into the <methods> section of a dependent data package.

See the emld library for more on working with EML as a list or JSON-LD. See the xml2 library for working with EML as XML.

Examples

## Not run:

methodStep <- get_provenance_metadata("knb-lter-pal.309.1")
methodStep
#> [1] <description>
 #> <para>This method step describes provenance-based ...
 #> [2] <dataSource>
 #> <title>Stable isotope composition (d18O) of seawat ...

## End(Not run)
get_recent_uploads

get_recent_uploads Get recent uploads

Description

Get recent uploads

Usage

get_recent_uploads(query, as = "data.frame", env = "production")

Arguments

query (character) Query (see details below)
as (character) Format of the returned object. Can be: "data.frame" or "xml".
env (character) Repository environment. Can be: "production", "staging", or "development".

Details

Query parameters are specified as key=value pairs, multiple pairs must be delimited with ampersands (&), and only a single value should be specified for a particular key. The following query parameter keys are allowed:

- serviceMethod - Can be: createDataPackage, updateDataPackage
- fromTime - An ISO8601 timestamp
- limit - A positive whole number

The query parameter serviceMethod should have the value "createDataPackage" (to retrieve recent inserts) or "updateDataPackage" (to retrieve recent updates). The query parameter fromTime is used to specify the date/time in the past that represents the oldest audit records that should be returned. Data packages uploaded prior to that time are not considered recent uploads and are thus filtered from the query results. The query parameter limit sets an upper limit on the number of audit records returned. For example, "limit=3".

Value

(data.frame or xml_document) A list of zero or more audit records of either recently inserted or recently updated data packages.

See Also

Other Audit Manager Services: get_audit_count(), get_audit_record(), get_audit_report(), get_docid_reads(), get_packageid_reads()
## is_authorized

### Examples

```
## Not run:

# Get the 5 most recently created data packages
auditReport <- get_recent_uploads(
    query = "serviceMethod=createDataPackage&limit=5"
)

## End(Not run)
```

### Description

Is authorized to read

### Usage

```
is_authorized(resourceId, env = "production")
```

### Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>resourceId</td>
<td>(character) Resource identifier</td>
</tr>
<tr>
<td>env</td>
<td>(character) Repository environment. Can be: &quot;production&quot;, &quot;staging&quot;, or &quot;development&quot;.</td>
</tr>
</tbody>
</table>

### Value

(logical) TRUE if the authenticated user has permission to read the specified resource

### Note

User authentication is required (see `login()`)

### See Also

Other Miscellaneous: `create_data_package_archive()`, `create_dn()`

### Examples

```
## Not run:

login()

# Get the most recently created data package
auditReport <- get_recent_uploads(
    query = "serviceMethod=createDataPackage&limit=1"
)
```
# Get the resourceId
resourceId <- xml2::xml_text(
  xml2::xml_find_all(auditReport, ".//resourceId")
)
resourceId
#> [1] "https://pasta.lternet.edu/package/eml/knb-lter-hbr/345/1"

# Check read authorization
is_authorized(resourceId)
#> [1] TRUE

logout()

## End(Not run)

---

---

list_active_reservations

List active reservations

Description
List active reservations

Usage
list_active_reservations(as = "data.frame", env = "production")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>as</td>
<td>(character) Format of the returned object. Can be: &quot;data.frame&quot; or &quot;xml&quot;.</td>
</tr>
<tr>
<td>env</td>
<td>(character) Repository environment. Can be: &quot;production&quot;, &quot;staging&quot;, or &quot;development&quot;.</td>
</tr>
</tbody>
</table>

Value
(data.frame or xml_document) The set of data package identifiers that users have actively reserved. Note that data package identifiers that have been successfully uploaded are no longer considered active reservations and thus are not included in this list.

See Also
Other Identifier Reservations: create_reservation(), delete_reservation(), list_reservation_identifiers()
list_data_descendants  List data descendants

Description

Data descendants are data packages that are known to be derived, in whole or in part, from the specified source data package.

Usage

list_data_descendants(packageId, as = "data.frame", env = "production")

Arguments

packageId  (character) Data package identifier
as         (character) Format of the returned object. Can be: "data.frame" or "xml".
env        (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Descendants of packageId

See Also

Other Listing: list_data_entities(), list_data_package_identifiers(), list_data_package_revisions(), list_data_package_scopes(), list_data_sources(), list_deleted_data_packages(), list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

## Not run:

# List descendants
dataDescendants <- list_data_descendants("knb-lter-bnz.501.17")

## End(Not run)
Description

List data entities

Usage

list_data_entities(packageId, env = "production")

Arguments

packageId (character) Data package identifier

env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(character) Identifiers for all data entities in packageId

See Also

Other Listing: list_data_descendants(), list_data_package_identifiers(), list_data_package_revisions(), list_data_package_scopes(), list_data_sources(), list_deleted_data_packages(), list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

## Not run:

dateIds <- list_data_entities("knb-lter-and.2732.7")
dateIds

#> [1] "0464a1d9262fc6e609cb0b24adb7e5ba"
#> [2] "cc3ade83d3655edd2ca67421a52ef46"

## End(Not run)
list_data_package_citations

List data package citations

Description

List data package citations

Usage

list_data_package_citations(
  packageId,
  as = "data.frame",
  list_all = FALSE,
  env = "production"
)

Arguments

packageId  (character) Data package identifier
as         (character) Format of the returned object. Can be: "data.frame" or "xml".
list_all   (logical) Return all citations within a data package series?
env        (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) A list of journal citations

See Also

Other Journal Citations: create_journal_citation(), delete_journal_citation(), get_journal_citation(),
list_principal_owner_citations()

Examples

## Not run:

# List citations
journalCitations <- list_data_package_citations("edi.845.1")

## End(Not run)
list_data_package_identifiers

List data package identifiers

Description

List data package identifiers

Usage

list_data_package_identifiers(scope, env = "production")

Arguments

  scope  (character) Scope of data package
  env    (character) Repository environment. Can be: "production", "staging", or "development".

Value

  (numeric) Identifiers of data packages within a specified scope

See Also

  Other Listing: list_data_descendants(), list_data_entities(), list_data_package_revisions(),
                 list_data_package_scopes(), list_data_sources(), list_deleted_data_packages(),
                 list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

  ## Not run:

  # List identifiers
  identifiers <- list_data_package_identifiers("knb-lter-ble")
  identifiers
  #> [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 23

  ## End(Not run)
list_data_package_revisions

List data package revisions

Description

List data package revisions

Usage

list_data_package_revisions(
  scope,
  identifier,
  filter = NULL,
  env = "production"
)

Arguments

  scope     (character) Scope of data package
  identifier  (numeric) Identifier of data package
  filter    (character) Filter results by "newest" or "oldest"
  env       (character) Repository environment. Can be: "production", "staging", or "development".

Value

  (numeric) Revisions of a data package within a specified scope and identifier

See Also

Other Listing: list_data_descendants(), list_data_entities(), list_data_package_identifiers(), list_data_package_scopes(), list_data_sources(), list_deleted_data_packages(), list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

  ## Not run:

  # List revisions
  revisions <- list_data_package_revisions("knb-lter-arc", 20131)
  revisions
  #> [1] 1 2

  ## End(Not run)
list_data_package_scopes

List data package scopes

Description
List data package scopes

Usage
list_data_package_scopes(env = "production")

Arguments

env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(numeric) Scopes within a specified env

See Also

Other Listing: list_data_descendants(), list_data_entities(), list_data_package_identifiers(),
list_data_package_revisions(), list_data_sources(), list_deleted_data_packages(),
list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

## Not run:

# List scopes
scopes <- list_data_package_scopes()
scopes

#> [1] "ecotrends" "edi" "knb-lter-and"
#> [4] "knb-lter-arc" "knb-lter-bes" "knb-lter-ble"
#> [7] "knb-lter-bnz" "knb-lter-cap" "knb-lter-ccce"
#> [10] "knb-lter-cdr" "knb-lter-cwt" "knb-lter-fce"
#> [13] "knb-lter-gce" "knb-lter-hbr" "knb-lter-hfr"
#> [16] "knb-lter-jrn" "knb-lter-kbs" "knb-lter-knz"
#> [19] "knb-lter-luq" "knb-lter-mcm" "knb-lter-mcr"
#> [22] "knb-lter-nes" "knb-lter-nin" "knb-lter-ntl"
#> [25] "knb-lter-nwk" "knb-lter-nwt" "knb-lter-pal"
#> [28] "knb-lter-pie" "knb-lter-sbc" "knb-lter-sev"
#> [31] "knb-lter-sgs" "knb-lter-vcr" "lter-landsat"
#> [34] "lter-landsat-ledaps" "msb-cap" "msb-paleon"
#> [37] "msb-tembiodev"

## End(Not run)
**list_data_sources**

**List data sources**

**Description**

Data sources are data packages, or other online digital objects, that are known to be inputs to the specified derived data package.

**Usage**

```r
list_data_sources(packageId, as = "data.frame", env = "production")
```

**Arguments**

- `packageId` (character): Data package identifier
- `as` (character): Format of the returned object. Can be: "data.frame" or "xml".
- `env` (character): Repository environment. Can be: "production", "staging", or "development".

**Details**

Data sources can be either internal or external to the EDI data repository. Internal data sources include a packageId value and a URL to the source metadata. For data sources external to PASTA, the packageId element will be empty and a URL value may or not be documented.

**Value**

(data.frame or xml_document) Data sources to packageId

**See Also**

Other Listing: `list_data_descendants()`, `list_data_entities()`, `list_data_package_identifiers()`, `list_data_package_revisions()`, `list_data_package_scopes()`, `list_deleted_data_packages()`, `list_recent_changes()`, `list_recent_uploads()`, `list_service_methods()`, `list_user_data_packages()`

**Examples**

```r
## Not run:
# List sources
dataSources <- list_data_sources("edi.275.4")

## End(Not run)
```
list_deleted_data_packages

List deleted data packages

Description

List deleted data packages

Usage

list_deleted_data_packages(env = "production")

Arguments

env  (character) Repository environment. Can be: "production", "staging", or "development".

Value

(character) All data packages (excluding revision values) that have been deleted from the data package registry.

See Also

Other Listing: list_data_descendants(), list_data_entities(), list_data_package_identifiers(), list_data_package_revisions(), list_data_package_scopes(), list_data_sources(), list_recent_changes(), list_recent_uploads(), list_service_methods(), list_user_data_packages()

Examples

## Not run:

# List deleted data packages
deleted <- list_deleted_data_packages()
head(deleted)
#> [1] "edi.10"  "edi.222"  "edi.419"  "edi.511"  "edi.857"  "edi.878"

## End(Not run)
list_principal_owner_citations

List principal owner citations

Description

List principal owner citations

Usage

list_principal_owner_citations(
  principalOwner,
  as = "data.frame",
  env = "production"
)

Arguments

principalOwner  (character) Principal owner in the format returned by create_dn()
as               (character) Format of the returned object. Can be: "data.frame" or "xml".
env              (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Journal citations metadata for all entries owned by the specified principal owner

See Also

Other Journal Citations: create_journal_citation(), delete_journal_citation(), get_journal_citation(), list_data_package_citations()

Examples

## Not run:

# List citations
dn <- create_dn(userId = "FCE", ou = "EDI")
journalCitations <- list_principal_owner_citations(principalOwner = dn)

## End(Not run)
Description

List all data package insert, update, and delete operations, optionally specifying the date and time to and/or from which the changes should be listed. An optional scope value can be specified to filter results for a particular data package scope.

Usage

```r
list_recent_changes(
  fromDate = NULL,
  toDate = NULL,
  scope = NULL,
  as = "data.frame",
  env = "production"
)
```

Arguments

- `fromDate` (character) Start date in the format "YYYY-MM-DDThh:mm:ss"
- `toDate` (character) End date in the format "YYYY-MM-DDThh:mm:ss"
- `scope` (character) Scope of data package
- `as` (character) Format of the returned object. Can be: "data.frame" or "xml".
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Recent changes and their corresponding packageId, scope, identifier, revision, principal, doi, serviceMethod, and date.

See Also

Other Listing: `list_data_descendants()`, `list_data_entities()`, `list_data_package_identifiers()`, `list_data_package_revisions()`, `list_data_package_scopes()`, `list_data_sources()`, `list_deleted_data_packages()`, `list_recent_uploads()`, `list_service_methods()`, `list_user_data_packages()`

Examples

```r
## Not run:
# Changes occurring in the first 3 days of 2021 for all scopes
dataPackageChanges <- list_recent_changes(
  fromDate = "2021-01-01T00:00:00",
  toDate = "2021-01-03T00:00:00"
)
```
list_recent_uploads

## End(Not run)

---

**list_recent_uploads**  List recent uploads

### Description

List recent uploads

### Usage

```r
list_recent_uploads(type, limit = 5, as = "data.frame", env = "production")
```

### Arguments

- **type**  (character) Upload type. Can be: "insert" or "update".
- **limit**  (numeric) Maximum number of results to return
- **as**  (character) Format of the returned object. Can be: "data.frame" or "xml".
- **env**  (character) Repository environment. Can be: "production", "staging", or "development".

### Value

(data.frame or xml_document) Data package uploads

### See Also

Other Listing: `list_data_descendants()`, `list_data_entities()`, `list_data_package_identifiers()`, `list_data_package_revisions()`, `list_data_package_scopes()`, `list_data_sources()`, `list_deleted_data_packages()`, `list_recent_changes()`, `list_service_methods()` , `list_user_data_packages()`

### Examples

```r
## Not run:

# Get the 3 newest revisions
dataPackageUploads <- list_recent_uploads("update", 3)

## End(Not run)
```
list_reservation_identifiers

List reservation identifiers

Description

List reservation identifiers

Usage

list_reservation_identifiers(scope, env = "production")

Arguments

- **scope** (character) Scope of data package
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

Value

(numeric) The set of identifiers for the specified scope that end users have actively reserved for future upload.

See Also

Other Identifier Reservations: `create_reservation()`, `delete_reservation()`, `list_active_reservations()`

Examples

```r
## Not run:
# List reservations
reservations <- list_reservation_identifiers(scope = "edi")
reservations
#> [1] 11 130 131 132 142 152 154 156 158 159 161 162 171
#> [14] 172 173 174 175 177 178 180 182 183 185 196 203 ...

## End(Not run)
```
list_service_methods  List service methods

Description
List service methods

Usage
list_service_methods(env = "production")

Arguments
env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(character) A simple list of web service methods supported by the Data Package Manager web service

See Also
Other Listing: list_data_descendants(), list_data_entities(), list_data_package_identifiers(), list_data_package_revisions(), list_data_package_scopes(), list_data_sources(), list_deleted_dataPackages(), list RecentChanges(), list RecentUploads(), list userDataPackages()

Examples
## Not run:
# All service methods
services <- list_service_methods()
services
#> [1] "appendProvenance"    "createDataPackage"
#> [3] "createDataPackageArchive"  "createReservation"
#> [5] "deleteReservation"     "deleteDataPackage"
#> [7] "evaluateDataPackage"   "getProvenanceMetadata"
#> [9] "isAuthorized"          "listActiveReservations"
#> [11] "listDataEntities"      "listDataDescendants"
#> [13] "listDataSources"       "listRecentChanges"
#> [15] "listDataPackageIdentifiers" "listDataPackageRevisions"
#> [17] "listDataPackageScopes" "listDeletedDataPackages"
#> [19] "listRecentUploads"     "listReservationIdentifiers"
#> [21] "listServiceMethods"    "listUserDataPackages"
#> [23] "listWorkingOn"         "readDataEntity"
#> [25] "readDataEntityAcl"     "readDataEntityRmd"
#> [27] "readDataEntityChecksum" "readDataEntityDoi"
#> [29] "readDataEntityName"    "readDataEntityNames"
list_user_data_packages

List user data packages

Description

List all data packages (including their revision values) uploaded to the repository by a particular user, specified by a distinguished name. Data packages that were uploaded by the specified user but have since been deleted are excluded from the list.

Usage

list_user_data_packages(dn, env = "production")

Arguments

  dn (character) Distinguished name of user. Create with create_dn().
  env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(character) Data package identifiers belonging to a dn

See Also

Other Listing: list_data_descendants(), list_data_entities(), list_data_package_identifiers(), list_data_package_revisions(), list_data_package_scopes(), list_data_sources(), list_deleted_data_packages(), list_recent_changes(), list_recent_uploads(), list_service_methods()
Examples

```r
## Not run:
# List user data packages
dn <- create_dn(userId = "dbjourneynorth")
packageIds <- list_user_data_packages(dn)
packageIds
#> [1] "edi.948.1" "edi.949.1"

## End(Not run)
```

### Description

List working on

### Usage

```r
list_working_on(as = "data.frame", env = "production")
```

### Arguments

- **as** (character) Format of the returned object. Can be: "data.frame" or "xml".
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

### Value

(data.frame or xml_document) The set of data packages the EDI repository is currently working on inserting or updating. Note that data packages currently being evaluated by the EDI repository are not included in the list.

### Examples

```r
## Not run:
list_working_on()

## End(Not run)
```
**Description**

Login to the EDI repository

**Usage**

```r
login(userId = NULL, userPass = NULL, config = NULL)
```

**Arguments**

- `userId` (character) User identifier of an EDI data repository account. If using
- `userPass` (character) Password of `userId`
- `config` (character) Path to config.txt, which contains `userId` and `userPass` (see details below)

**Details**

If `userId`, `userPass`, and `config` are `NULL`, the console will prompt for credentials.

`config`: Supplying credentials in a file named config.txt facilitates authentication within automated/unassisted processes. Contents of this file should be new line separated and have the form "<argument> = <value>" (e.g. `userId = myname`).

**Value**

(character) A temporary (~10 hour) authentication token written to the system variable "EDI_TOKEN".

**Note**

Only works when authenticating with EDI credentials. Does not work when authenticating with ORCID, GitHub, or Google credentials.

Be careful not to accidentally share your `userId` and `userPass`. Some tips to avoid this:

- Don’t write code that explicitly lists your credentials.
- Don’t save your workspace when exiting an R session.
- Do store your credentials as environmental variables and reference these.
- Do use `config` but if using version control ensure the config.txt file is listed in your .gitignore.

If you may have shared your credentials, please reset your password at [https://dashboard.edirepository.org/dashboard/auth/reset_password_init](https://dashboard.edirepository.org/dashboard/auth/reset_password_init).

**See Also**

Other Authentication: `logout()`
Examples

## Not run:

# Interactively at the console
login()
#> User name: "my_name"
#> User password: "my_secret"

# Programmatically with function arguments
login(userId = "my_name", userPass = "my_secret")

# Programmatically with a file containing userId and userPass arguments
login(config = paste0(tempdir(), "/config.txt"))

## End(Not run)

### logout

#### Logout of the EDI repository

Description

Logout of the EDI repository

Usage

logout()

Details

Removes the temporary authentication token system variable "EDI_TOKEN".

Value

(NULL) No return value.

See Also

Other Authentication: login()

Examples

## Not run:

logout()

## End(Not run)
query_event_subscriptions

Query event subscriptions

Description

Query event subscriptions

Usage

query_event_subscriptions(query = NULL, as = "data.frame", env = "production")

Arguments

query  (character) Query (see details below)
as     (character) Format of the returned object. Can be: "data.frame" or "xml".
env    (character) Repository environment. Can be: "production", "staging", or "development".

Details

Query parameters are specified as key=value pairs, multiple pairs must be delimited with ampersands (&), and only a single value should be specified for a particular key. The following query parameter keys are allowed:

- creator
- scope
- identifier
- revision
- url

If a query parameter is specified, and a subscription’s respective attribute does not match it, that subscription will not be included in the group of subscriptions returned. If scope, identifier, or revision are used, their values must together constitute a syntactically and semantically correct EML packageId (i.e. "scope.identifier.revision") - either partial or complete. If url is used, its value must not contain ampersands. Therefore, if a subscription’s URL contains ampersands, it cannot be filtered based on its URL.

Value

(data.frame or xml_document) A list of the subscriptions whose attributes match those specified in the query string (see details below). If a query string is omitted, all subscriptions in the subscription database will be returned for which the requesting user is authorized to read. If query parameters are included, they are used to filter that set of subscriptions based on their attributes.
Note

User authentication is required (see `login()`)

See Also

Other Event Notifications: `create_event_subscription()`, `delete_event_subscription()`, `execute_event_subscription()`, `get_event_subscription_schema()`, `get_event_subscription()`

Examples

```r
## Not run:

login()

# Query subscriptions
query <- "scope=edi"
subscriptions <- query_event_subscriptions(query, env = "staging")

logout()

## End(Not run)
```

---

**read_data_entity**

Read data entity

**Description**

Read data entity

**Usage**

```r
read_data_entity(packageId, entityId, env = "production")
```

**Arguments**

- `packageId` (character) Data package identifier
- `entityId` (character) Data entity identifier
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

**Value**

(raw) Raw bytes (i.e. application/octet-stream) to be parsed by a reader function appropriate for the data type
read_data_entity

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples

## Not run:

```r
# Read names and IDs of data entities in package "edi.1047.1"
res <- read_data_entity_names(packageId = "edi.1047.1")
res
#> entityId entityName
#> 1 3abac5f99eccc1585879178a355176f6d Environmentals.csv
#> 2 f6bf89b48ced8292840e53567cbf0c8 ByCatch.csv
#> 3 c75642ddcb430132b4b1a86bede996 Chinook.csv
#> 4 2c9ee86cc3f3f729c5f8be0a2a1d Steelhead.csv
#> 5 785690848dd20f491063725009c6819 TrapEfficiencyRelease.csv
#> 6 58b900439a5671ea7fe13212e889ba5 TrapEfficiencySummary.csv
#> 7 86e61c1a501b7dcf040d18e009bf87 TrapOperations.csv

# Read raw bytes of the 3rd data entity
raw <- read_data_entity(packageId = "edi.1047.1", entityId = res$entityId[3])
head(raw)
#> [1] ef bb bf 44 61 74

# Parse with .csv reader
data <- readr::read_csv(file = raw)
data
#> # A tibble: 105,325 x 20
#> # ... with 105,315 more rows, and 13 more variables:
#> # atCaptureRun <chr>, finalRun <chr>, finalRunMethod <chr>,
#> # lifeStage <chr>, forkLength <dbl>, weight <dbl>, mort <chr>
```
read_data_entity_checksum

Description
Read data entity checksum

Usage
read_data_entity_checksum(packageId, entityId, env = "production")

Arguments
- packageId (character) Data package identifier
- entityId (character) Data entity identifier
- env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(character) A 40-character SHA-1 checksum value of entityId in packageId

See Also
Other Accessing: read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples
## Not run:

## List data entities
entityIds <- list_data_entities(packageId = "knb-lter-ble.1.7")
entityIds
#> [1] "a1723e0e5f3c4881f1a7ede1b036aba6"

read_data_entity_checksum

Read data entity checksum

Description
Read data entity checksum

Usage
read_data_entity_checksum(packageId, entityId, env = "production")

Arguments
- packageId (character) Data package identifier
- entityId (character) Data entity identifier
- env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(character) A 40-character SHA-1 checksum value of entityId in packageId

See Also
Other Accessing: read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples
## Not run:

## List data entities
entityIds <- list_data_entities(packageId = "knb-lter-ble.1.7")
entityIds
#> [1] "a1723e0e5f3c4881f1a7ede1b036aba6"
Read data entity name

Description
Read data entity name

Usage
read_data_entity_name(packageId, entityId, env = "production")

Arguments
- packageId: (character) Data package identifier
- entityId: (character) Data entity identifier
- env: (character) Repository environment. Can be: "production", "staging", or "development".

Value
(character) Name of entityId in packageId

See Also
Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()
Examples

```r
## Not run:
# List entities
entityIds <- list_data_entities(packageId = "knb-lter-cap.691.2")
entityIds
#> [1] "f6e4ef0b04aea3860724824ca05c5dd"
#> [2] "d225d480e75cc7888b41928682cda4c6"
#> [3] "d5cb834556480e48636157e4dee49e"

# Read name
entityName <- read_data_entity_name(
    packageId = "knb-lter-cap.691.2",
    entityId = entityIds[1]
)
entityName
#> [1] "691_arthropods_00742cd00ab0d3d02337e28d1c919654.csv"

## End(Not run)
```

read_data_entity_names

**Read data entity names**

Description

Read data entity names

Usage

`read_data_entity_names(packageId, env = "production")`

Arguments

- `packageId` (character): Data package identifier
- `env` (character): Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame) Names and identifiers of all data entities in `packageId`

See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`
### Not run:

```r
read_data_entity_names("knb-lter-cap.691.2")
```

```r
#> entityId
#> 1 f6e4efd0b04aea3860724824ca05c5dd
#> 2 d2263480e75cc7888b41928602cda4c6
#> 3 d5cb83e4556408e48f636157e4deee49e
```

```r
#> entityName
#> 1 691_arthropods_00742cd00ab0d3d02337e28d1c919654.csv
#> 2 691_captures_e5f57a98ae0b7941b10d4a600645495a.csv
#> 3 691_sampling_events_e8d76d7e76385e4ae84bcafb754d0093.csv
```

```r
## End(Not run)
```

### Description

Read data entity resource metadata

### Usage

```r
read_data_entity_resource_metadata(
  packageId,  
  entityId,  
  as = "data.frame",  
  env = "production"  
)
```

### Arguments

- `packageId` (character) Data package identifier
- `entityId` (character) Data entity identifier
- `as` (character) Format of the returned object. Can be: "data.frame" or "xml".
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

### Value

(data.frame or xml_document) The resource metadata of `entityId` in `packageId`
### read_data_entity_size

**Read data entity size**

#### Description

Read data entity size

#### Usage

```r
read_data_entity_size(packageId, entityId, env = "production")
```

#### Arguments

- `packageId` (character): Data package identifier
- `entityId` (character): Data entity identifier
- `env` (character): Repository environment. Can be: "production", "staging", or "development".

#### Examples

```r
## Not run:

# List entities
entityIds <- list_data_entities(packageId = "knb-lter-cce.310.1")
head(entityIds)
#> [1] "4aaff61e0d316130be0b445d3013877"
#> [2] "088775341e7fb65206af8c967d076e2"
#> [3] "6982dd880eba66470c492af3dc08f2459"
#> [4] "782fbaa20ea62987c838378e9eadcfa6"
#> [5] "ae8ecd148df1275b30358577d0fa6b4a"
#> [6] "a53b312efe0a176fdfe74ab7cc0916b"

# Read resource metadata for first entity
resourceMetadata <- read_data_entity_resource_metadata(
  packageId = "knb-lter-cce.310.1",
  entityId = entityIds[1]
)

## End(Not run)
```
**Value**

(numeric) Size, in bytes, of entityId in packageId

**See Also**

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

**Examples**

```r
## Not run:

# List data entities
entityIds <- list_data_entities(packageId = "knb-lter-cdr.711.1")
entityIds
#> [1] "c61703839eac9a641ea0c3c69dc3345b"

# Read size
size <- read_data_entity_size(
  packageId = "knb-lter-cdr.711.1",
  entityId = entityIds
)
size
#> [1] 707094

## End(Not run)
```

---

read_data_entity_sizes

*Read data entity sizes*

**Description**

Read data entity sizes

**Usage**

```r
read_data_entity_sizes(packageId, env = "production")
```
read_data_package

Description

Read data package

Usage

read_data_package(packageId, ore = FALSE, env = "production")
**Arguments**

- **packageId**  (character) Data package identifier
- **ore**  (logical) Return an OAI-ORE compliant resource map in RDF-XML format
- **env**  (character) Repository environment. Can be: "production", "staging", or "development".

**Value**

(character or xml_document) A resource map with reference URLs to each of the metadata, data, and quality report resources that comprise the packageId.

**See Also**

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_evaluate_report()`, `read_metadata()`

**Examples**

```r
## Not run:
# Get resource map
resourceMap <- read_data_package(packageId = "knb-lter-cwt.5026.13")
resourceMap
#> [1] "https://pasta.lternet.edu/package/data/eml/knb-lter-cwt/5026/13/ ... 
#> [2] "https://pasta.lternet.edu/package/data/eml/knb-lter-cwt/5026/13/ ... 
#> [3] "https://pasta.lternet.edu/package/metadata/eml/knb-lter-cwt/5026 ... 

# Get resource map in ORE format
resourceMap <- read_data_package(  
  packageId = "knb-lter-cwt.5026.13",  
  ore = TRUE  
)
resourceMap
#> {xml_document}
#> <RDF xmlns:cito="http://purl.org/spar/cito" xmlns:dc="http://purl.or ... 
#> [1] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [2] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [3] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [4] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [5] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [6] <rdf:Description rdf:about="https://pasta.lternet.edu/package/eml ... 
#> [7] <rdf:Description rdf:about="http://environmentaldatainitiative.or ... 
#> [8] <rdf:Description rdf:about="http://www.openarchives.org/ore/terms ... 
```
read_data_package_archive

Read data package archive

Description
Read data package archive

Usage
read_data_package_archive(packageId, transaction, path, env = "production")

Arguments
- packageId (character) Data package identifier
- transaction (character) Transaction identifier. This parameter is DEPRECATED.
- path (character) Path of directory in which the result will be written
- env (character) Repository environment. Can be: "production", "staging", or "development".

Value
(.zip file) The data package archive of packageId requested by transaction

See Also
Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples
## Not run:
# Download zip archive
read_data_package_archive("knb-lter-sev.31999.1", path = tempdir())
## [1] "knb-lter-sev.31999.1.zip"
Description

Read data package citation

Usage

```r
read_data_package_citation(
  packageId,
  access = TRUE,
  style = "ESIP",
  ignore = NULL,
  as = "char",
  env = "production"
)
```

Arguments

- `packageId` (character): Data package identifier
- `access` (logical): Return a datestamp in the citation of the current UTC date. This is recommended by the ESIP citation style guide.
- `style` (character): Set the style for which to format the citation. Can be: "ESIP", "DRYAD", "BIBTEX", "RAW".
- `ignore` (character): Ignore individuals, organizations, or positions in the author list. Can be: "INDIVIDUALS", "ORGANIZATIONS", or "POSITIONS". See details below.
- `as` (character): Format of the returned citation. Can be: "char", "html", "json".
- `env` (character): Repository environment. Can be: "production", "staging", or "development".

Details

A citation may consist of a list of authors, publication year, title, data package version, publisher, digital object identifier, and access date. The order and presence of these components depends on the style requested for the citation (see query parameters above).

A brief discussion of the fields in a citation:
• Authors - This function uses content extracted from the science metadata described by an Ecological Metadata Language (EML) document to generate the author list. Specifically, it uses the creator section of EML to generate the list of authors, including individuals, organizations, and positions.

This function preserves the order of the creator list as defined within the EML document. As such, if you would like the citation to begin with an organization name, you should position the creator element that describes the organization at the beginning of the creator list in the EML document.

This function also assumes that a creator element contains information pertaining to only a single "creator", although EML allows for multiple identities in a single creator element. It will do its best to accommodate multi-named subjects within a creator element, but mileage will vary.

This function is opinionated in how it determines an author: individuals, take precedence over organizations and positions, and organizations take precedence over positions. What this means is if an individual and organization and position are all defined in a single creator element, this function sets the author to the named information within the individual element; and, if only an organization and position exist within a single creator element, this function will set the author to the named information within the organization element. Finally, if only a position is defined within a single creator element, this function will set the author to the named information within the position element. It is important to note that this function respects the creator content as defined in the EML document and will set a position name to an author if it is present and meets the above hierarchy. If you believe that a position should not be displayed as data package author, then you should not include it as a data package creator.

Finally, this function does not collect or use tertiary information (e.g., phone number, addresses, emails) from within the creator element since this type of information is not used as part of a data package citation.

• Publication Year - The publication year is defined by the calendar year when the data package was archived into the EDI data repository. The publication year may differ from the year of the publication date entered into the EML, which is often set to the date when the data package became publicly available, although not yet archived into the EDI data repository.

• Title - This function uses the title section of EML as the citation title. EML title elements are copied verbatim into the citation.

• Version Number - The citation version number represents the revision step (or increment) of the data package as archived in the EDI data repository. Revision values are whole numbers and have a one-to-one correspondence to the revision of the data package in the repository.

• Publisher - By default, the publisher field of the citation is permanently set to "Environmental Data Initiative". This value will not change during the tenure of the EDI data repository.

• DOI - The Digital Object Identifier (DOI) is the EDI generated DOI value that is registered with DataCite, and is displayed using the fully qualified "doi.org" URL. This DOI URL will resolve to the corresponding "landing page" of the data package as displayed on the EDI Data Portal.

• Access Date - The access date is the UTC date in which the citation was requested.

Value

(character or html_document or json) The data package citation
See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`,
`read_data_entity_resource_metadata()`, `read_data_entity_sizes()`,
`read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`,
`read_data_package_doi()`, `read_data_package_error()`,
`read_data_package_from_doi()`, `read_data_package_report_checksum()`,
`read_data_package_report_resource_metadata()`,
`read_data_package_report_summary()`, `read_data_package_report()`,
`read_data_package_resource_metadata()`,
`read_data_package()`, `read_evaluate_report_summary()`,
`read_evaluate_report()`, `read_metadata_checksum()`,
`read_metadata_dublin_core()`, `read_metadata_entity()`,
`read_metadata_format()`, `read_metadata_resource_metadata()`,
`read_metadata()`

Examples

```R
## Not run:

packageId <- "edi.460.1"

# Retrieve "ESIP" stylized citation (default) in plain text format
citation <- read_data_package_citation(packageId)
citation

# Retrieve "DRYAD" stylized citation in plain text format
citation <- read_data_package_citation(packageId, style = "DRYAD")
citation

# Retrieve "ESIP" stylized citation (default) in HTML format
citation <- read_data_package_citation(packageId, as = "html")
citation
#> {html_document}
#> <html>

# Retrieve "ESIP" stylized citation (default), ignoring individuals, in
# plain text format
citation <- read_data_package_citation(packageId, ignore = "INDIVIDUALS")
citation
#> [1] "Texas A&M University at Galveston, Texas A&M University - Corpu ..."

## End(Not run)
```

---

**read_data_package_doi**  
Read data package Digital Object Identifier

**Description**

Read data package Digital Object Identifier
read_data_package_doi

Usage

```r
read_data_package_doi(packageId, as_url = FALSE, env = "production")
```

Arguments

- `packageId` (character) Data package identifier
- `as_url` (logical) Returns the DOI as a URL if TRUE.
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(character) The Digital Object Identifier for `packageId`

See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_data_package()`.

Examples

```r
## Not run:
# Read package DOI
doi <- read_data_package_doi("knb-lter-jrn.210548103.15")
doi
#> [1] "doi:10.6073/pasta/c80c0c03d22791524d4b870d2193c843"

# Read package DOI as URL
doi <- read_data_package_doi("knb-lter-jrn.210548103.15", as_url = TRUE)
doi
#> [1] "https://doi.org/10.6073/pasta/c80c0c03d22791524d4b870d2193c843"

## End(Not run)
```

read_data_package_error

Description

Read data package error
read_data_package_from_doi

Usage

read_data_package_error(transaction, env = "production")

Arguments

transaction (character) Transaction identifier
env (character) Repository environment. Can be: "production", "staging", or "development".

Value

An error is returned if an error occurred while processing the request, otherwise NULL is returned if no error was encountered or if processing is still underway.

Note

User authentication is required (see login())

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()
Value

(character or xml_document) A resource map with reference URLs to each of the metadata, data, and quality report resources that comprise the data package.

See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_data_package()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_metadata()`

Examples

```r
## Not run:

# Get resource map
resourceMap <- read_data_package_from_doi(
  doi = "doi:10.6073/pasta/b202c11db7c64943f6b4ed9f8c17fb25"
)
resourceMap
#> [1] "https://pasta.lternet.edu/package/data/eml/knb-lter-fce/1233/2/5 ...

# Get resource map in ORE format
resourceMap <- read_data_package_from_doi(
  doi = "doi:10.6073/pasta/b202c11db7c64943f6b4ed9f8c17fb25",
  ore = TRUE
)
resourceMap
#> <xml_document>
#> <RDF xmlns:cito="http://purl.org/spar/cito" xmlns:dc="http://purl.or ...
Description

Read data package report

Usage

read_data_package_report(packageId, as = "xml", env = "production")

Arguments

packageId (character) Data package identifier
as (character) Format of the returned report. Can be: "xml", "html", or "char".
env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(xml_document) Data package report

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report_summary(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples

## Not run:

# Read as XML
qualityReport <- read_data_package_report("knb-lter-knz.260.4")
qualityReport
#> <xml_document> ...
#> [1] <creationDate>2020-02-04T16:38:38</creationDate>
#> [2] <packageId>knb-lter-knz.260.4</packageId>
#> [3] <includeSystem>lter</includeSystem>
#> [4] <includeSystem>knb</includeSystem>
#> <qualityCheck qualityType="metadata" system="...
## Description

Read data package report checksum

## Usage

```
read_data_package_report_checksum(packageId, env = "production")
```

## Arguments

- **packageId** (character) Data package identifier
- **env** (character) Repository environment. Can be: "production", "staging", or "development".

## Value

(character) A 40 character SHA-1 checksum value for the report
See Also
Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()` , `read_data_package_resource_metadata()`, `read_data_package()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_metadata()`.

Examples
## Not run:

```r
# Read report checksum
packageId <- "knb-lter-luq.208.1"
checksum <- read_data_package_report_checksum(packageId)
checksum
#> "980dbf3f3cd739533b711b005722033bdcd12f"

## End(Not run)
```

---

### read_data_package_report_resource_metadata

**Read data package report resource metadata**

#### Description
Read data package report resource metadata

#### Usage
```r
read_data_package_report_resource_metadata(
  packageId,
  as = "data.frame",
  env = "production"
)
```

#### Arguments

- `packageId` *(character)* Data package identifier
- `as` *(character)* Format of the returned object. Can be: "data.frame" or "xml".
- `env` *(character)* Repository environment. Can be: "production", "staging", or "development".

#### Value
*(data.frame or xml_document)* Report resource metadata
read_data_package_report_summary

Summarize the data package quality report

Description

Summarize the data package quality report

Usage

read_data_package_report_summary(
  packageId,
  with_exceptions = TRUE,
  env = "production"
)

Arguments

  packageId    (character) Data package identifier
  with_exceptions    (logical) Convert quality report warnings and errors to R warnings and errors
  env    (character) Repository environment. Can be: "production", "staging", or "development".

Examples

## Not run:

# Read resource metadata
resourceMetadata <- read_data_package_report_resource_metadata(
  packageId = "knblter-mcm.9129.3"
)

## End(Not run)
Value

(message/warning/error) A message listing the total number of checks resulting in valid, info, warn, and error status. Exceptions are raised if warnings and errors are found and with_exceptions is TRUE.

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(), read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(), read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(), read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(), read_data_package_report_resource_metadata(), read_data_package_report(), read_data_package_resource_metadata(), read_data_package(), read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples

## Not run:

```r
# Read report summary
read_data_package_report_summary("knb-lter-knz.260.4")
#> ===================================================
#> EVALUATION REPORT
#> ==============================================================
#> PackageId: knb-lter-knz.260.4
#> Report Date/Time: 2020-02-04T16:38:38
#> Total Quality Checks: 213
#> Valid: 139
#> Info: 73
#> Warn: 1
#> Error: 0
#> Warning message:
#> One or more quality checks resulted in 'warn'
#>
#> # End(Not run)
```

---

### Description

Read data package resource metadata
Usage

```r
read_data_package_resource_metadata(
  packageId,
  as = "data.frame",
  env = "production"
)
```

Arguments

- `packageId` (character) Data package identifier
- `as` (character) Format of the returned object. Can be: "data.frame" or "xml".
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Resource metadata of `packageId`

See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_metadata()`

Examples

```r
## Not run:

# Read resource metadata
resourceMetadata <- read_data_package_resource_metadata(
  packageId = "edi.613.1"
)

## End(Not run)
```

Description

Read evaluate report
Usage

read_evaluate_report(transaction, as = "xml", env = "production")

Arguments

  transaction (character) Transaction identifier
  as (character) Format of the returned report. Can be: "xml", "html", or "char".
  env (character) Repository environment. Can be: "production", "staging", or "development".

Value

(xml_document or html_document or character) The evaluate quality report document

Note

User authentication is required (see login())

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(),
read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(),
read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(),
read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(),
read_data_package_report_resource_metadata(), read_data_package_report_summary(),
read_data_package_report(), read_data_package_resource_metadata(), read_data_package(),
read_evaluate_report_summary(), read_metadata_checksum(), read_metadata_dublin_core(),
read_metadata_entity(), read_metadata_format(), read_metadata_resource_metadata(),
read_metadata()

Examples

```r
## Not run:

login()

# Evaluate data package
transaction <- evaluate_data_package(
  eml = paste0(tempdir(), "edi.595.1.xml"),
  env = "staging"
)
transaction
#> [1] "evaluate_163966785813042760"

# Read as HTML and write to file for a web browser view
qualityReport <- read_evaluate_report(
  transaction = transaction,
  as = "html",
  env = "staging"
)
```

writeLines(qualityReport, paste0(tempdir(), "/report.html"))

# Read as character and write to file for browsing
qualityReport <- read_evaluate_report(
  transaction = transaction,
  as = "char",
  env = "staging"
)
writeLines(qualityReport, paste0(tempdir(), "/report.txt"))

# Read as XML
qualityReport <- read_evaluate_report(
  transaction = transaction,
  env = "staging"
)
qualityReport

logout()

## End(Not run)

---

**read_evaluate_report_summary**

*Summarize the evaluate quality report*

**Description**

Summarize the evaluate quality report

**Usage**

```r
read_evaluate_report_summary(
  transaction,  
  with_exceptions = TRUE,  
  env = "production"
)
```

**Arguments**

- `transaction` (character) Transaction identifier
with_exceptions

(logical) Convert quality report warnings and errors to R warnings and errors

env

(character) Repository environment. Can be: "production", "staging", or "development".

Details

Get transaction from evaluate_data_package()

Value

(message/warning/error) A message listing the total number of checks resulting in valid, info, warn, and error status. Exceptions are raised if warnings and errors are found and with_exceptions is TRUE.

Note

User authentication is required (see login())

See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name().
read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(),
read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(),
read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(),
read_data_package_report_resource_metadata(), read_data_package_report_summary(),
read_data_package_report(), read_data_package_resource_metadata(), read_data_package(),
read_evaluate_report(), read_metadata_checksum(), read_metadata_dublin_core(), read_metadata_entity(),
read_metadata_format(), read_metadata_resource_metadata(), read_metadata()

Examples

```r
## Not run:

login()

# Evaluate data package
transaction <- evaluate_data_package(
  eml = paste0(tempdir(), "/edi.595.1.xml"),
  env = "staging"
)
transation
#> [1] "evaluate_163966785813042760"

# Summarize report
read_evaluate_report_summary(transaction, env = "staging")
#> ===================================================
#> EVALUATION REPORT
#> ===================================================
#>
```
# PackageId: edi.595.1
# Report Date/Time: 2021-12-16T22:49:25
# Total Quality Checks: 29
# Valid: 21
# Info:  8
# Warn: 0
# Error: 0

logout()

## End(Not run)

## read_metadata

<table>
<thead>
<tr>
<th>read_metadata</th>
<th>Read metadata</th>
</tr>
</thead>
</table>

### Description

Read metadata

### Usage

```r
read_metadata(packageId, env = "production")
```

### Arguments

| packageId | (character) Data package identifier |
| env       | (character) Repository environment. Can be: "production", "staging", or "development". |

### Value

(xml_document) EML metadata document.

See the [emld](https://cran.r-project.org/package=emld) library for more on working with EML as a list or JSON-LD. See the [xml2](https://cran.r-project.org/package=xml2) library for working with EML as XML.

### See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_data_package()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`
Examples

```r
## Not run:

# Read metadata
eml <- read_metadata("edi.100.1")
eml
#> {xml_document}
#> <eml packageId="edi.100.1" system="https://pasta.edirepository.org" ... 
#> [1] <access authSystem="https://pasta.edirepository.org/authentication ... 
#> [2] <dataset>
 # alternateIdentifier system="https://doi.org">doi:10 ... 

## End(Not run)
```

---

**read_metadata_checksum**

*Read metadata checksum*

### Description

Read metadata checksum

### Usage

```
read_metadata_checksum(packageId, env = "production")
```

### Arguments

- `packageId` (character) Data package identifier
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

### Value

(character) A 40 character SHA-1 checksum value

### See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`.

- `read_data_package_report()`
- `read_data_package_resource_metadata()`
- `read_data_package()`
- `read_evaluate_report_summary()`
- `read_evaluate_report()`
- `read_metadata_dublin_core()`
- `read_metadata_entity()`
- `read_metadata_format()`
- `read_metadata_resource_metadata()`
- `read_metadata()`
### Examples

```r
## Not run:
# Read checksum
checksum <- read_metadata_checksum("knb-lter-ntl.409.1")
checksum
## End(Not run)
```

---

**read_metadata_dublin_core**

*Read metadata Dublin Core*

#### Description

Read metadata Dublin Core

#### Usage

```r
read_metadata_dublin_core(packageId, env = "production")
```

#### Arguments

- `packageId` *(character)* Data package identifier
- `env` *(character)* Repository environment. Can be: "production", "staging", or "development".

#### Value

*(xml_document)* Dublin Core metadata.

See the *xml2* library for more on working with XML.

#### See Also

Other Accessing: `read_data_entity_checkum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_data_package()`, `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_entity()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_metadata()`
Examples

```r
## Not run:

# Read dc metadata
dc <- read_metadata_dublin_core("knb-lter-nes.10.1")
dc
#> {xml_document}
#> <dc schemaLocation="http://www.openarchives.org/OAI/2.0/oai_dc/ http: ...#
#> [1] <dc:type/>
#> [2] <dc:identifier/>

## End(Not run)
```

---

**read_metadata_entity**  
Read data entity metadata

### Description
Read data entity metadata

### Usage

```r
read_metadata_entity(packageId, entityId, env = "production")
```

### Arguments

- `packageId`  
  (character) Data package identifier

- `entityId`  
  (character) Data entity identifier

- `env`  
  (character) Repository environment. Can be: "production", "staging", or "development".

### Value

(xml_nodeset) The metadata of `entityId` in `packageId`

### See Also

Other Accessing: `read_data_entity_checksum()`, `read_data_entity_names()`, `read_data_entity_name()`, `read_data_entity_resource_metadata()`, `read_data_entity_sizes()`, `read_data_entity_size()`, `read_data_entity()`, `read_data_package_archive()`, `read_data_package_citation()`, `read_data_package_doi()`, `read_data_package_error()`, `read_data_package_from_doi()`, `read_data_package_report_checksum()`, `read_data_package_report_resource_metadata()`, `read_data_package_report_summary()`, `read_data_package_report()`, `read_data_package_resource_metadata()`, `read_data_package()` , `read_evaluate_report_summary()`, `read_evaluate_report()`, `read_metadata_checksum()`, `read_metadata_dublin_core()`, `read_metadata_format()`, `read_metadata_resource_metadata()`, `read_metadata()`
Examples

```r
# Read entity names and IDs
packageId <- "knb-lter-cap.691.2"
entities <- read_data_entity_names(packageId)
entities
#> entityId
#> 1 f6e4efd0b04aea3860724824ca05c5dd
#> 2 d2263480e75cc7888b41928602cda4c6
#> 3 d5cb83e4556408e48f636157e4dee49e
#> entityName
#> 1 691_arthropods_00742cd00ab0d3d02337e28d1c919654.csv
#> 2 691_captures_e5f57a98ae0b7941b10d4a606645495a.csv
#> 3 691_sampling_events_e8d76e76385e4ae84bcafb754d0093.csv

# Read metadata of the first entity
meta <- read_metadata_entity(packageId, entityId = entities$entityId[1])
meta
#> {xml_nodeset (1)}
#> [1] <dataTable id="691_arthropods_00742cd00ab0d3d02337e28d1c919654.cs ... 
```

## End(Not run)

read_metadata_format  Read metadata format

Description

Read metadata format

Usage

`read_metadata_format(packageId, env = "production")`

Arguments

- `packageId` (character) Data package identifier
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(character) Metadata format type
See Also

Other Accessing: read_data_entity_checksum(), read_data_entity_names(), read_data_entity_name(),
read_data_entity_resource_metadata(), read_data_entity_sizes(), read_data_entity_size(),
read_data_entity(), read_data_package_archive(), read_data_package_citation(), read_data_package_doi(),
read_data_package_error(), read_data_package_from_doi(), read_data_package_report_checksum(),
read_data_package_report_resource_metadata(), read_data_package_report_summary(),
read_data_package_report(), read_data_package_resource_metadata(), read_data_package(),
read_evaluate_report_summary(), read_evaluate_report(), read_metadata_checksum(),
read_metadata_dublin_core(), read_metadata_entity(), read_metadata_resource_metadata(),
read_metadata()

Examples

## Not run:

```r
# Read format
dataFormat <- read_metadata_format("knb-lter-nwt.930.1")

#> [1] "eml://ecoinformatics.org/eml-2.1.1"

# End(Not run)
```

read_metadata_resource_metadata

Read metadata resource metadata

Description

Read metadata resource metadata

Usage

```r
read_metadata_resource_metadata(
  packageId,
  as = "data.frame",
  env = "production"
)
```

Arguments

- `packageId` (character) Data package identifier
- `as` (character) Format of the returned object. Can be: "data.frame" or "xml".
- `env` (character) Repository environment. Can be: "production", "staging", or "development".

Value

(data.frame or xml_document) Resource metadata for the data package metadata resource
search_data_packages

Search data packages

Description

Searches data packages in the EDI data repository using the specified Solr query.

Usage

search_data_packages(query, as = "data.frame", env = "production")

Arguments

query (character) Query (see details below)

as (character) Format of the returned object. Can be: "data.frame" or "xml".

env (character) Repository environment. Can be: "production", "staging", or "development".

Details

Documents in the EDI data repository Solr index can be discovered based on metadata values stored in the following list of searchable fields (not all EML content is queryable):

Single-value fields:

- abstract
- begindate - In ISO format (YYYY-MM-DDThh:mm:ss)

Examples

```r
## Not run:

# Read resource metadata
resourceMetadata <- read_metadata_resource_metadata(
  packageId = "knb-lter-pal.309.1"
)
## End(Not run)
```
• doi
• enddate - In ISO format (YYYY-MM-DDThh:mm:ss)
• funding
• geographicdescription
• id
• methods
• packageid - Data Id in "scope.identifier.revision" format
• pubdate - In ISO format (YYYY-MM-DDThh:mm:ss)
• responsibleParties
• scope
• singledate
• site
• taxonomic
• title

Multi-value fields:
• author
• coordinates - Use "IsWithin(West+East+North+South)" where each cardinal direction is in decimal degrees with South of the equator as negative and East of the prime meridian positive.
• keyword
• organization
• projectTitle
• relatedProjectTitle
• timescale

query parser: The optimal query parser (defType=edismax) is added to every query.
See Apache Solr Wiki for how to construct a Solr query.

Value

(data.frame or xml_document) Search results containing the fields:
• abstract
• begindate
• doi
• enddate
• funding
• geographicdescription
• id
• methods
search_data_packages

- packageid
- pubdate
- responsibleParties
- scope
- site
- taxonomic
- title
- authors
- spatialCoverage
- sources
- keywords
- organizations
- singledates
- timescales

Note

Only the newest version of data packages are searchable, older versions are not.

When constructing a query note that the 15403 data packages of the ecotrends project and 10492 data packages of the LTER Landsat project, can be excluded from the returned results by including &fq=-scope:(ecotrends+lter-landsat) in the query string.

Examples

## Not run:

```r
# Search for data packages containing the term "air temperature"
res <- search_data_packages(query = 'q="air+temperature"&fl=*')

# Search for data packages containing the term "air temperature" and
# returning only the packageid, title, and score of each match
res <- search_data_packages(query = 'q="air+temperature"&fl=packageid,title,score')

# Search for data packages containing the term "air temperature", returning
# only the packageid, title, score, and excluding ecotrends and lter-landsat
# scopes from the returned results
query <- paste0('q="air+temperature"&fl=packageid,title, score&',
                'fq=-scope:(ecotrends+lter-landsat)')
res <- search_data_packages(query)

## End(Not run)
update_data_package  Update data package

Description
Update data package

Usage
update_data_package(eml, useChecksum = FALSE, env = "production")

Arguments
eml  (character) Full path to an EML file describing the data package to be updated
useChecksum  (logical) Use data entities from a previous version of the data package? See
details below.
env  (character) Repository environment. Can be: "production", "staging", or "develop-
ment".

Details
Each data entity described in eml must be accompanied by a web accessible URL at the XPath
"//physical/distribution/online/url". The EDI data repository uses these links to download the data
entities. The URLs must be static and not have any redirects otherwise the data entities will not be
downloadable.

Value
transaction  (character) Transaction identifier. May be used in a subsequent call to check_status_update()
to determine the operation status

Note
User authentication is required (see login())

See Also
Other Evaluation and Upload: check_status_create(), check_status_evaluate(), check_status_update(),
create_data_package(), evaluate_data_package()

Examples
## Not run:

login()

# Update data package
transaction <- update_data_package(
update_data_package

eml = paste0(tempdir(), "/edi.595.2.xml"),
env = "staging"
)
transaction
#> [1] "update_edi.595_163966788658131920__edi.595.2"

# Check update status
status <- check_status_update(
  transaction = transaction,
  env = "staging"
)
status
#> [1] TRUE

logout()

## End(Not run)
Index

* Accessing
  read_data_entity, 52
  read_data_entity_checksum, 54
  read_data_entity_name, 55
  read_data_entity_names, 56
  read_data_entity_resource_metadata, 57
  read_data_entity_size, 58
  read_data_entity_sizes, 59
  read_data_package, 60
  read_data_package_archive, 62
  read_data_package_citation, 63
  read_data_package_doi, 65
  read_data_package_error, 66
  read_data_package_from_doi, 67
  read_data_package_report, 69
  read_data_package_report_checksum, 70
  read_data_package_report_resource_metadata, 71
  read_data_package_report_summary, 72
  read_data_package_resource_metadata, 73
  read_evaluate_report, 74
  read_evaluate_report_summary, 76
  read_metadata, 78
  read_metadata_checksum, 79
  read_metadata_dublin_core, 80
  read_metadata_entity, 81
  read_metadata_format, 82
  read_metadata_resource_metadata, 83
* Audit Manager Services
  get_audit_count, 21
  get_audit_record, 22
  get_audit_report, 23
  get_docid_reads, 25
  get_packageid_reads, 29
* Authentication
  login, 49
  logout, 50
* Browse and Discovery
  search_data_packages, 84
* Evaluation and Upload
  check_status_create, 3
  check_status_evaluate, 5
  check_status_update, 6
  create_data_package, 7
  evaluate_data_package, 17
  update_data_package, 87
* Event Notifications
  create_event_subscription, 10
  delete_event_subscription, 14
  execute_event_subscription, 19
  get_event_subscription, 26
  get_event_subscription_schema, 27
  query_event_subscriptions, 51
* Identifier Reservations
  create_reservation, 13
  delete_reservation, 16
  list_active_reservations, 33
  list_reservation_identifiers, 45
* Journal Citations
  create_journal_citation, 11
  delete_journal_citation, 15
  get_journal_citation, 28
  list_data_package_citations, 36
  list_principal_owner_citations, 42
* Listing
  list_data_descendants, 34
  list_data_entities, 35
  list_data_package_identifiers, 37
  list_data_package_revisions, 38
  list_data_package_scopes, 39
  list_data_sources, 40
  list_deleted_data_packages, 41
  get_recent_uploads, 31
list_recent_changes, 43
list_recent_uploads, 44
list_service_methods, 46
list_user_data_packages, 47

* Miscellaneous
  create_data_package_archive, 9
  create_dn, 9
  is_authorized, 32

* Provenance
  get_provenance_metadata, 30

* System Monitoring
  list_working_on, 48

    check_status_create, 3, 5, 7, 8, 18, 87
    check_status_evaluate, 4, 5, 7, 8, 18, 87
    check_status_update, 4, 5, 6, 8, 18, 87
    create_data_package, 4, 5, 7, 8, 18, 87
    create_data_package_archive, 9, 10, 32
    create_dn, 9, 9, 32
    create_event_subscription, 10, 14, 20, 26, 27, 52
    create_journal_citation, 11, 16, 28, 36, 42
    create_reservation, 13, 17, 33, 45

    delete_event_subscription, 10, 14, 20, 26, 27, 52
    delete_journal_citation, 12, 15, 28, 36, 42
    delete_reservation, 13, 16, 33, 45

    evaluate_data_package, 4, 5, 7, 8, 17, 87
    execute_event_subscription, 10, 14, 19, 26, 27, 52

    get_audit_count, 21, 23–25, 29, 31
    get_audit_record, 22, 22, 24, 25, 29, 31
    get_audit_report, 22, 23, 23, 25, 29, 31
    get_docid_reads, 22–24, 25, 29, 31
    get_event_subscription, 10, 14, 20, 26, 27, 52
    get_event_subscription_schema, 10, 14, 20, 26, 27, 52
    get_journal_citation, 12, 16, 28, 36, 42
    get_packageid_reads, 22–25, 29, 31
    get_provenance_metadata, 30
    get_recent_uploads, 22–23, 29, 31

    is_authorized, 9, 10, 32

    list_active_reservations, 13, 17, 33, 45
    list_data_descendants, 34, 35, 37–41, 43, 44, 46, 47
    list_data_entities, 34, 35, 37–41, 43, 44, 46, 47
    list_data_package_citations, 12, 16, 28, 36, 42
    list_data_package_identifiers, 34, 35, 37, 38–41, 43, 44, 46, 47
    list_data_package_revisions, 34, 35, 37, 38, 39–41, 43, 44, 46, 47
    list_data_package_scopes, 34, 35, 37, 38, 39, 40, 41, 43, 44, 46, 47
    list_data_sources, 34, 35, 37–39, 40, 41, 43, 44, 46, 47
    list_deleted_data_packages, 34, 35, 37–40, 41, 43, 44, 46, 47
    list_principal_owner_citations, 12, 16, 28, 36, 42
    list_recent_changes, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_uploads, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_uploads, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_changes, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_uploads, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_uploads, 34, 35, 37–41, 43, 44, 46, 47
    list_recent_uploads, 34, 35, 37–41, 43, 44, 46, 47
    login, 49, 50
    logout, 49, 50

    query_event_subscriptions, 10, 14, 20, 26, 27, 51

    read_data_entity_name, 53, 54, 55, 56, 58–62, 65–69, 71–75, 77–81, 83, 84
    read_data_entity_size, 53–56, 58, 58, 60–62, 65–69, 71–75, 77–81, 83, 84
INDEX

read_data_entity_sizes, 53–56, 58, 59, 59, 61, 62, 65–69, 71–75, 77–81, 83, 84
read_data_package_report_summary, 53–55, 57–62, 65–69, 71, 72, 72, 74, 75, 77–81, 83, 84
read_metadata_format, 53–55, 57–62, 65–69, 71–75, 77–81, 82, 84
search_data_packages, 84
update_data_package, 4, 5, 7, 8, 18, 87