

# Package ‘rDataPipeline’

October 14, 2022

**Title** Functions to Interact with the 'FAIR Data Pipeline'

**Version** 0.54.1

**Description** R implementation of the 'FAIR Data Pipeline API'. The 'FAIR Data Pipeline' is intended to enable tracking of provenance of FAIR (findable, accessible and interoperable) data used in epidemiological modelling.

**License** GPL (>= 3)

**Imports** assertthat, cli, configr, dplyr, git2r, httr, jsonlite, openssl, R6, rhdf5, semver, stats, usethis, utils, yaml

**Suggests** units, testthat

**biocViews** rhdf5

**Encoding** UTF-8

**RoxygenNote** 7.1.2

**URL** <https://www.fairdatapipeline.org/rDataPipeline/>,  
<https://github.com/FAIRDataPipeline/rDataPipeline>

**BugReports** <https://github.com/FAIRDataPipeline/rDataPipeline/issues>

**NeedsCompilation** no

**Author** Sonia Mitchell [cre, aut] (<<https://orcid.org/0000-0003-1536-2066>>),  
Ryan Field [ctb] (<<https://orcid.org/0000-0002-4424-9890>>)

**Maintainer** Sonia Mitchell <[sonia.mitchell@glasgow.ac.uk](mailto:sonia.mitchell@glasgow.ac.uk)>

**Repository** CRAN

**Date/Publication** 2021-11-17 21:00:06 UTC

## R topics documented:

rDataPipeline-package	3
add_read	4
add_write	5
check_config	6
check_dataproduct_exists	6
check_datetime	7

check_exists . . . . .	8
check_field . . . . .	8
check_fields . . . . .	9
check_handle . . . . .	9
check_integer . . . . .	10
check_local_repo . . . . .	10
check_string . . . . .	11
check_table_exists . . . . .	11
check_yaml_write . . . . .	12
clean_query . . . . .	12
create_config . . . . .	13
create_index . . . . .	13
create_version_number . . . . .	14
download_from_database . . . . .	14
download_from_url . . . . .	15
extract_id . . . . .	16
fair_init . . . . .	16
fair_run . . . . .	17
fdp-class . . . . .	17
fdp_resolve_read . . . . .	22
fdp_resolve_write . . . . .	22
finalise . . . . .	23
findme . . . . .	23
find_read_match . . . . .	24
find_write_match . . . . .	24
get_author_url . . . . .	25
get_components . . . . .	25
get_dataproduct . . . . .	26
get_entity . . . . .	26
get_entry . . . . .	27
get_existing . . . . .	27
get_fields . . . . .	28
get_file_hash . . . . .	29
get_github_hash . . . . .	29
get_id . . . . .	30
get_index . . . . .	30
get_max_version . . . . .	31
get_storage_location . . . . .	31
get_token . . . . .	31
get_url . . . . .	32
increment_filename . . . . .	32
initialise . . . . .	33
is_queryable . . . . .	33
link_read . . . . .	34
link_write . . . . .	34
new_author . . . . .	35
new_code_repo_release . . . . .	35
new_code_run . . . . .	36

new_data_product . . . . .	37
new_external_object . . . . .	38
new_file_type . . . . .	39
new_issue . . . . .	39
new_keyword . . . . .	40
new_licence . . . . .	41
new_namespace . . . . .	41
new_object . . . . .	42
new_object_component . . . . .	43
new_quality_controlled . . . . .	44
new_storage_location . . . . .	44
new_storage_root . . . . .	45
new_user_author . . . . .	46
paper_exists . . . . .	46
raise_issue . . . . .	47
raise_issue_config . . . . .	47
raise_issue_repo . . . . .	48
raise_issue_script . . . . .	48
random_hash . . . . .	49
read_array . . . . .	49
read_distribution . . . . .	50
read_estimate . . . . .	50
read_table . . . . .	51
register_issue_dataproduct . . . . .	51
register_issue_script . . . . .	52
remove_empty_parents . . . . .	52
resolve_read . . . . .	53
resolve_version . . . . .	53
resolve_write . . . . .	54
validate_fields . . . . .	54
write_array . . . . .	55
write_distribution . . . . .	56
write_estimate . . . . .	57
write_table . . . . .	57

<b>Index</b>	<b>59</b>
--------------	-----------

---

rDataPipeline-package *rDataPipeline*

---

## Description

FAIR Data Pipeline API

## Details

For more information see <https://www.fairdatapipeline.org/>

---

`add_read`*add\_read*

---

**Description**

Add data product to read block of user-written config file. Used in combination with `create_config()` for unit testing.

**Usage**

```
add_read(  
  path,  
  data_product,  
  component,  
  version,  
  use_data_product,  
  use_component,  
  use_version,  
  use_namespace  
)
```

**Arguments**

<code>path</code>	config file path
<code>data_product</code>	data_product field
<code>component</code>	component field
<code>version</code>	(optional) version field
<code>use_data_product</code>	(optional) use_data_product field
<code>use_component</code>	(optional) use_component field
<code>use_version</code>	(optional) use_version field
<code>use_namespace</code>	(optional) use_namespace field

**Examples**

```
## Not run:  
path <- "test_config/config.yaml"  
  
# Write run_metadata block  
create_config(path = path,  
              description = "test",  
              input_namespace = "test_user",  
              output_namespace = "test_user")  
  
# Write read block  
add_read(path = path,
```

```

        data_product = "test/array",
        component = "level/a/s/d/f/s",
        version = "0.2.0")

## End(Not run)

```

---

add\_write

*add\_write*


---

## Description

Add data product to read block of user-written config file. Used in combination with `create_config()` for unit testing.

## Usage

```

add_write(
  path,
  data_product,
  description,
  version,
  file_type,
  use_data_product,
  use_component,
  use_version,
  use_namespace
)

```

## Arguments

path	config file path
data_product	data_product field
description	component field
version	(optional) version field
file_type	(optional) file type field
use_data_product	(optional) use_data_product field
use_component	(optional) use_component field
use_version	(optional) use_version field
use_namespace	(optional) use_namespace field

**Examples**

```
## Not run:
path <- "test_config/config.yaml"

# Write run_metadata block
create_config(path = path,
              description = "test",
              input_namespace = "test_user",
              output_namespace = "test_user")

# Write read block
add_write(path = path,
          data_product = "test/array",
          description = "data product description",
          version = "0.2.0")

## End(Not run)
```

---

check_config	<i>check_config</i>
--------------	---------------------

---

**Description**

check\_config

**Usage**

```
check_config(handle, data_product, what)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	data_product
what	what

---

check_dataproduct_exists	<i>check_dataproduct_exists</i>
--------------------------	---------------------------------

---

**Description**

If a data product already exists with the same name, version, and namespace, throw an error

**Usage**

```
check_dataproduct_exists(  
    write_dataproduct,  
    write_version,  
    write_namespace_id,  
    endpoint  
)
```

**Arguments**

```
write_dataproduct      write_dataproduct  
write_version          write_version  
write_namespace_id     write_namespace_id  
endpoint              endpoint
```

---

check_datetime	<i>check_datetime</i>
----------------	-----------------------

---

**Description**

check\_datetime

**Usage**

```
check_datetime(table, this_field, query_class, this_query)
```

**Arguments**

```
table          a string specifying the name of the table  
this_field     a string specifying the name of the field  
query_class    a string specifying the class of the field  
this_query     a string specifying the contents of the field
```

---

check_exists	<i>Check if entry exists in the data registry</i>
--------------	---------------------------------------------------

---

**Description**

Check whether an entry already exists in a table (in the data registry)

**Usage**

```
check_exists(table, query)
```

**Arguments**

table	a string specifying the name of the table
query	a list containing a valid query for the table, e.g. list(field = value)

**Value**

Returns TRUE if the entry exists and FALSE if it doesn't

---

check_field	<i>check_field</i>
-------------	--------------------

---

**Description**

check\_field

**Usage**

```
check_field(table, this_field, query_class, this_query, method, endpoint)
```

**Arguments**

table	a string specifying the name of the table
this_field	a string specifying the name of the field
query_class	a string specifying the class of the field
this_query	a string specifying the contents of the field
method	a string specifying the method, c("GET", "POST")
endpoint	endpoint



---

check_fields	<i>check_fields</i>
--------------	---------------------

---

**Description**

check\_fields

**Usage**

check\_fields(table, query, method, endpoint)

**Arguments**

table	a string specifying the name of the table
query	a list containing the query
method	a string specifying the method, c("GET", "POST")
endpoint	endpoint

---

check_handle	<i>check_handle</i>
--------------	---------------------

---

**Description**

check\_handle

**Usage**

check\_handle(handle, data\_product, what, component)

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
what	element in handle – one of c("inputs", "outputs")
component	a string specifying the name of the component

---

check_integer	<i>check_integer</i>
---------------	----------------------

---

**Description**

check\_integer

**Usage**

check\_integer(table, this\_field, query\_class, this\_query)

**Arguments**

table	a string specifying the name of the table
this_field	a string specifying the name of the field
query_class	a string specifying the class of the field
this_query	a string specifying the contents of the field

---

check_local_repo	<i>check_local_repo</i>
------------------	-------------------------

---

**Description**

check\_local\_repo

**Usage**

check\_local\_repo(path)

**Arguments**

path	Local repository file path
------	----------------------------

**Value**

boolean, if local repository is clean (TRUE, else FALSE)

---

check_string	<i>check_string</i>
--------------	---------------------

---

**Description**

check\_string

**Usage**

check\_string(table, this\_field, this\_query)

**Arguments**

table	a string specifying the name of the table
this_field	a string specifying the name of the field
this_query	a string specifying the contents of the field

---

check_table_exists	<i>Check if table exists</i>
--------------------	------------------------------

---

**Description**

Check if table exists in the data registry

**Usage**

check\_table\_exists(table)

**Arguments**

table	a string specifying the name of the table
-------	-------------------------------------------

**Value**

Returns TRUE if a table exists, FALSE if it doesn't

---

check_yaml_write	<i>check_yaml_write</i>
------------------	-------------------------

---

**Description**

check\_yaml\_write

**Usage**

```
check_yaml_write(handle, data_product)
```

**Arguments**

handle	fdp object
data_product	a string specifying the name of the data product

---

clean_query	<i>Clean query</i>
-------------	--------------------

---

**Description**

Function to clean a query and return it without an api prefix

**Usage**

```
clean_query(data, endpoint)
```

**Arguments**

data	a list containing a valid query for the table, e.g. list(field = value)
endpoint	endpoint

---

create_config	<i>create_config</i>
---------------	----------------------

---

**Description**

Generates (user generated) config.yaml files for unit tests. Use add\_read() and add\_write() functions to add read and write blocks.

**Usage**

```
create_config(
    path,
    description,
    input_namespace,
    output_namespace,
    write_data_store = file.path(tempdir(), "datastore", ""),
    force = TRUE,
    local_repo = "local_repo"
)
```

**Arguments**

path	config file path
description	description field
input_namespace	input_namespace field
output_namespace	output_namespace field
write_data_store	write_data_store field
force	force
local_repo	local_repo

---

create_index	<i>create_index</i>
--------------	---------------------

---

**Description**

create\_index

**Usage**

```
create_index(self)
```

**Arguments**

self	self
------	------

---

create\_version\_number *Create version number*

---

### Description

Creates a version number from either *a date and a version* **or** *a date and major and patch* **or** *major minor patch*. If no parameters are supplied a default version is returned 0.1.0 This function prioritizes download date and version over all other parameters

### Usage

```
create_version_number(
  download_date = NULL,
  version = NULL,
  major = 0,
  minor = "1",
  patch = 0
)
```

### Arguments

download_date	(Optional) download_date This can either be a date or datetime but must include the full year <i>e.g</i> from Sys.Date() 2020-01-01 Or from Sys.time() 2020-01-01 00:00:00 BST note: also accepts / delimited dates <i>e.g</i> 01/02/2020 or 2020/01/01 and accepts date without delimiters but assumes ddmmyyyy or yyyymmdd <i>e.g.</i> 20200201
version	version number using major.minor.patch numbering <i>e.g.</i> 0.1.0 or major.patch <i>e.g.</i> 0.0
major	major number if not using version
minor	minor number if not using date
patch	patch number if not using version

### Value

returns a character vector in the format of major.minor.patch *e.g.* 0.20200101.0

---

download\_from\_database

*Download source file from database*

---

### Description

Download source file from database

**Usage**

```
download_from_database(  
    source_root,  
    source_path,  
    path,  
    filename,  
    overwrite = FALSE  
)
```

**Arguments**

source_root	a string specifying the source root
source_path	a string specifying the source path
path	a string specifying where the file will be saved
filename	a string specifying the filename (the name given to the saved file)
overwrite	a boolean specifying whether or not the file should be overwritten if it already exists

**See Also**

Other download functions: [download\\_from\\_url\(\)](#)

---

download_from_url	<i>Download source file from URL</i>
-------------------	--------------------------------------

---

**Description**

This function will download a file from a url

**Usage**

```
download_from_url(source_root, source_path, path, filename)
```

**Arguments**

source_root	a string specifying the source root
source_path	a string specifying the source path
path	a string specifying where the file will be saved
filename	a string specifying the filename (the name given to the saved file)

**See Also**

Other download functions: [download\\_from\\_database\(\)](#)

---

extract_id	<i>extract_id</i>
------------	-------------------

---

**Description**

extract\_id

**Usage**

```
extract_id(url, endpoint)
```

**Arguments**

url	url
endpoint	endpoint

---

fair_init	<i>fair_init</i>
-----------	------------------

---

**Description**

fair\_init

**Usage**

```
fair_init(name, identifier, endpoint = "http://localhost:8000/api/")
```

**Arguments**

name	a string specifying the full name or organisation name of the author; note that at least one of name or identifier must be specified
identifier	(optional) a string specifying the full URL identifier ( <i>e.g.</i> ORCID or ROR ID) of the author
endpoint	a string specifying the registry endpoint



---

fair_run	<i>fair_run</i>
----------	-----------------

---

**Description**

fair\_run

**Usage**

```
fair_run(
  path = "config.yaml",
  endpoint = "http://localhost:8000/api/",
  skip = FALSE
)
```

**Arguments**

path	string
endpoint	a string specifying the registry endpoint
skip	don't bother checking whether the repo is clean

---

fdp-class	<i>fdp-class</i>
-----------	------------------

---

**Description**

fdp-class  
fdp-class

**Details**

Container for class fdp

**Public fields**

yaml a list containing the contents of the working config.yaml  
 fdp\_config\_dir a string specifying the directory passed from fair\_run  
 model\_config a string specifying the URL of an entry in the object table associated with the storage\_location of the working config.yaml  
 submission\_script a string specifying the URL of an entry in the object table associated with the storage\_location of the submission script  
 code\_repo a string specifying the URL of an entry in the object table associated with the GitHub repository

code\_run a string specifying the URL of an entry in the code\_run table  
 inputs a data.frame containing metadata associated with code\_run inputs  
 outputs a data.frame containing metadata associated with code\_run outputs  
 issues a data.frame containing metadata associated with code\_run issues

## Methods

### Public methods:

- `fdp$new()`
- `fdp$print()`
- `fdp$input()`
- `fdp$output()`
- `fdp$output_index()`
- `fdp$raise_issue()`
- `fdp$finalise_output_hash()`
- `fdp$finalise_output_url()`
- `fdp$clone()`

**Method** `new()`: Create a new fdp object

*Usage:*

```
fdp$new(
  yaml,
  fdp_config_dir,
  model_config,
  submission_script,
  code_repo,
  code_run
)
```

*Arguments:*

yaml a list containing the contents of the working config.yaml  
 fdp\_config\_dir a string specifying the directory passed from fair run  
 model\_config a string specifying the URL of an entry in the object table associated with the storage\_location of the working config.yaml  
 submission\_script a string specifying the URL of an entry in the object table associated with the storage\_location of the submission script  
 code\_repo a string specifying the URL of an entry in the object table associated with the GitHub repository  
 code\_run a string specifying the URL of an entry in the code\_run table

*Returns:* Returns a new fdp object

**Method** `print()`: Print method

*Usage:*

```
fdp$print(...)
```

**Method** input(): Record code\_run inputs in fdp object

*Usage:*

```
fdp$input(  
  data_product,  
  use_data_product,  
  use_component,  
  use_version,  
  use_namespace,  
  path,  
  component_url  
)
```

*Arguments:*

data\_product a string specifying the name of the data product, used as a reference

use\_data\_product a string specifying the name of the data product, used as input in the code\_run

use\_component a string specifying the name of the data product component, used as input in the code\_run

use\_version a string specifying the data product version, used as input in the code\_run

use\_namespace a string specifying the namespace in which the data product resides, used as input in the code\_run

path a string specifying the location of the data product in the local data store

component\_url a string specifying the URL of an entry in the object\_component table

*Returns:* Returns an updated fdp object

**Method** output(): Record code\_run outputs in fdp object

*Usage:*

```
fdp$output(  
  data_product,  
  use_data_product,  
  use_component,  
  use_version,  
  use_namespace,  
  path,  
  data_product_description,  
  component_description,  
  public  
)
```

*Arguments:*

data\_product a string specifying the name of the data product, used as a reference

use\_data\_product a string specifying the name of the data product, used as output in the code\_run

use\_component a string specifying the name of the data product component, used as output in the code\_run

use\_version a string specifying the version of the data product, used as output in the code\_run

use\_namespace a string specifying the namespace in which the data product resides, used as output in the code\_run

path a string specifying the location of the data product in the local data store

data\_product\_description a string containing a description of the data product

component\_description a string containing a description of the data product component

public

*Returns:* Returns an updated fdp object

**Method** output\_index(): Return index of data product recorded in fdp object so that an issue may be attached

*Usage:*

```
fdp$output_index(data_product, component, version, namespace)
```

*Arguments:*

data\_product a string specifying the name of the data product, used as output in the code\_run

component a string specifying the name of the data product component, used as output in the code\_run

version a string specifying the name of the data product version, used as output in the code\_run

namespace a string specifying the namespace in which the data product resides, used as input in the code\_run

*Returns:* Returns an index used to identify the data product

**Method** raise\_issue(): Record issue in fdp object

*Usage:*

```
fdp$raise_issue(
  index,
  type,
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  issue,
  severity
)
```

*Arguments:*

index a numeric index, used to identify each input and output in the fdp object

type a string specifying the type of issue (one of "data", "config", "script", "repo")

use\_data\_product a string specifying the name of the data product, used as output in the code\_run

use\_component a string specifying the name of the data product component, used as output in the code\_run

use\_version a string specifying the name of the data product version, used as output in the code\_run

use\_namespace a string specifying the namespace in which the data product resides, used as input in the code\_run

issue a string containing a free text description of the issue

severity an integer specifying the severity of the issue

*Returns:* Returns an updated fdp object

**Method** finalise\_output\_hash(): Record file hash and update path name in fdp object

*Usage:*

```
fdp$finalise_output_hash(
  use_data_product,
  use_data_product_runid,
  use_version,
  use_namespace,
  hash,
  new_path,
  data_product_url,
  delete_if_duplicate = FALSE
)
```

*Arguments:*

use\_data\_product a string specifying the name of the data product, used as output in the code\_run

use\_data\_product\_runid a string specifying the name of the data product, the same as use\_data\_product excluding the RUN\_ID variable

use\_version a string specifying the name of the data product version, used as output in the code\_run

use\_namespace a string specifying the namespace in which the data product resides, used as input in the code\_run

hash a string specifying the hash of the file

new\_path a string specifying the updated location (filename is now the hash of the file) of the data product in the local data store

data\_product\_url a string specifying the URL of an object associated with the data\_product

delete\_if\_duplicate (optional) default is FALSE

*Returns:* Returns an updated fdp object

**Method** finalise\_output\_url(): Record data\_product and component URLs in fdp object

*Usage:*

```
fdp$finalise_output_url(
  use_data_product,
  use_component,
  use_version,
  use_namespace,
  component_url
)
```

*Arguments:*

use\_data\_product a string specifying the name of the data product, used as output in the code\_run

use\_component a string specifying the name of the data product component, used as output in the code\_run  
 use\_version a string specifying the name of the data product version, used as output in the code\_run  
 use\_namespace a string specifying the namespace in which the data product resides, used as input in the code\_run  
 component\_url a string specifying the URL of an entry in the object\_component table  
*Returns:* Returns an updated fdp object

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

*Usage:*

fdp\$clone(deep = FALSE)

*Arguments:*

deep Whether to make a deep clone.

fdp\_resolve\_read      *fdp\_resolve\_read*

### Description

fdp\_resolve\_read

### Usage

fdp\_resolve\_read(this\_read, yaml)

### Arguments

this_read	this_read
yaml	user written config file

fdp\_resolve\_write      *fdp\_resolve\_write*

### Description

fdp\_resolve\_write

### Usage

fdp\_resolve\_write(this\_write, yaml)

### Arguments

this_write	this_write
yaml	user written config file

---

finalise	<i>Finalise code run</i>
----------	--------------------------

---

**Description**

Finalise Code Run and push associated metadata to the local registry.

**Usage**

```
finalise(handle, delete_if_empty = FALSE, delete_if_duplicate = FALSE)
```

**Arguments**

handle	an object of class <code>fdp</code> , R6 containing metadata required by the Data Pipeline API
delete_if_empty	(optional) default is FALSE; see Details
delete_if_duplicate	(optional) default is FALSE; see Details

**Details**

If a Code Run does not read an input, write an output, or attach an issue, then delete the Code Run entry when `delete_if_empty` is set to TRUE.

If a data product has the same hash as a previous version, remove it from the registry when `delete_if_duplicate` is set to TRUE.

---

findme	<i>findme</i>
--------	---------------

---

**Description**

Returns metadata associated with the calculated hash of a target file. When multiple entries exist in the data registry all are returned.

**Usage**

```
findme(file, endpoint)
```

**Arguments**

file	file path
endpoint	endpoint

---

<code>find_read_match</code>	<i>Find matching read aliases in config file</i>
------------------------------	--------------------------------------------------

---

**Description**

Find read aliases in working config that match wildcard string

**Usage**

```
find_read_match(handle, data_product)
```

**Arguments**

<code>handle</code>	an object of class <code>fdp</code> , R6 containing metadata required by the Data Pipeline API
<code>data_product</code>	a string specifying the data product name

---

<code>find_write_match</code>	<i>Find matching write aliases in config file</i>
-------------------------------	---------------------------------------------------

---

**Description**

Find write aliases in working config that match wildcard string

**Usage**

```
find_write_match(handle, data_product)
```

**Arguments**

<code>handle</code>	an object of class <code>fdp</code> , R6 containing metadata required by the Data Pipeline API
<code>data_product</code>	a string specifying the data product name



---

<code>get_author_url</code>	<i>get_author_url</i>
-----------------------------	-----------------------

---

**Description**

`get_author_url`

**Usage**

`get_author_url(endpoint)`

**Arguments**

`endpoint`      a string specifying the registry endpoint

---

<code>get_components</code>	<i>Get H5 file components</i>
-----------------------------	-------------------------------

---

**Description**

Returns the names of the items at the root of the file

**Usage**

`get_components(filename)`

**Arguments**

`filename`      a string specifying a filename

**Value**

Returns the names of the items at the root of the file

**See Also**

Other get functions: [get\\_entry\(\)](#), [get\\_existing\(\)](#), [get\\_file\\_hash\(\)](#), [get\\_github\\_hash\(\)](#)

---

get_dataproduct	<i>get_dataproduct</i>
-----------------	------------------------

---

**Description**

get\_dataproduct

**Usage**

```
get_dataproduct(  
    data_product,  
    version,  
    namespace,  
    endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

data_product	data_product
version	version
namespace	namespace
endpoint	endpoint

---

get_entity	<i>Get entity from url</i>
------------	----------------------------

---

**Description**

Get entity from url

**Usage**

```
get_entity(url)
```

**Arguments**

url	a string specifying the url of an entry
-----	-----------------------------------------

---

get_entry	<i>Return all fields associated with a table entry in the data registry</i>
-----------	-----------------------------------------------------------------------------

---

**Description**

Return all fields associated with a table entry in the data registry

**Usage**

```
get_entry(table, query, endpoint = "http://localhost:8000/api/")
```

**Arguments**

table	a string specifying the name of the table
query	a list containing a valid query for the table, e.g. list(field = value)
endpoint	a string specifying the registry endpoint

**Value**

Returns a list of fields present in the specified entry

**See Also**

Other get functions: [get\\_components\(\)](#), [get\\_existing\(\)](#), [get\\_file\\_hash\(\)](#), [get\\_github\\_hash\(\)](#)

---

get_existing	<i>Return all entries posted to a table in the data registry</i>
--------------	------------------------------------------------------------------

---

**Description**

Get entries (from the data registry) in a particular table

**Usage**

```
get_existing(  
  table,  
  limit_results = TRUE,  
  detail = "all",  
  endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

table	a string specifying the name of the table
limit_results	a boolean specifying whether or not to limit the results, default is TRUE
detail	a string specifying what level of detail to return; options are "all" for all details or "id" for just URL and IDs
endpoint	a string specifying the registry endpoint

**Value**

Returns a `data.frame` of entries in table, default is limited to 100 entries

**See Also**

Other get functions: [get\\_components\(\)](#), [get\\_entry\(\)](#), [get\\_file\\_hash\(\)](#), [get\\_github\\_hash\(\)](#)

---

get_fields	<i>Get fields from table</i>
------------	------------------------------

---

**Description**

Use API endpoint to produce a list of fields for a table. Requires API key.

**Usage**

```
get_fields(table, endpoint = "http://localhost:8000/api/")
```

**Arguments**

table	a string specifying the name of the table
endpoint	a string specifying the registry endpoint

**Value**

Returns a `data.frame` of fields and their attributes set to "none"

---

get_file_hash	<i>Calculate hash from file</i>
---------------	---------------------------------

---

**Description**

Returns the SHA1 hash of a given file

**Usage**

```
get_file_hash(filename)
```

**Arguments**

filename            a string specifying a filename

**See Also**

Other get functions: [get\\_components\(\)](#), [get\\_entry\(\)](#), [get\\_existing\(\)](#), [get\\_github\\_hash\(\)](#)

---

get_github_hash	<i>Get current GitHub hash</i>
-----------------	--------------------------------

---

**Description**

Get the hash of the latest commit in the master branch of a particular repository. This function assumes git is installed and located in the System PATH.

**Usage**

```
get_github_hash(repo)
```

**Arguments**

repo                a string specifying the github username/repository

**See Also**

Other get functions: [get\\_components\(\)](#), [get\\_entry\(\)](#), [get\\_existing\(\)](#), [get\\_file\\_hash\(\)](#)

---

get_id	<i>Get ID</i>
--------	---------------

---

**Description**

Retrieve IDs for particular entries or all entries in a table

**Usage**

```
get_id(table, query = list(), endpoint = "http://localhost:8000/api/")
```

**Arguments**

table	a string specifying the name of the table
query	a list containing a valid query for the table, e.g. list(field = value)
endpoint	a string specifying the registry endpoint

**Value**

Returns a string or list of strings specifying the URL or URLs of entries in a table

---

get_index	<i>get_index</i>
-----------	------------------

---

**Description**

get\_index

**Usage**

```
get_index(write, data_product)
```

**Arguments**

write	write
data_product	data_product

---

get_max_version	<i>get_max_version</i>
-----------------	------------------------

---

**Description**

If entry doesn't exist in the registry, return version 0.0.0

**Usage**

```
get_max_version(data_product, namespace_id)
```

**Arguments**

data_product	data_product
namespace_id	namespace_id

---

get_storage_location	<i>Get storage location from url</i>
----------------------	--------------------------------------

---

**Description**

Get storage location entry

**Usage**

```
get_storage_location(location)
```

**Arguments**

location	the url of an entry in the storage_location table
----------	---------------------------------------------------

**Value**

Returns a list of fields associated with the specified entry

---

get_token	<i>get_token</i>
-----------	------------------

---

**Description**

get\_token

**Usage**

```
get_token()
```

---

get_url	<i>Get URL</i>
---------	----------------

---

**Description**

Retrieve URLs for particular entries or all entries in a table

**Usage**

```
get_url(table, query = list(), endpoint = "http://localhost:8000/api/")
```

**Arguments**

table	a string specifying the name of the table
query	a list containing a valid query for the table, <i>e.g.</i> list(field = value)
endpoint	a string specifying the registry endpoint

**Value**

Returns a string or list of strings specifying the URL or URLs of entries in a table

---

increment_filename	<i>increment_filename</i>
--------------------	---------------------------

---

**Description**

Searches directory for duplicate files and increments filename.

**Usage**

```
increment_filename(path)
```

**Arguments**

path	path
------	------



---

initialise	<i>Initialise code run</i>
------------	----------------------------

---

**Description**

Reads in a working config file, generates new Code Run entry, and returns a handle containing various metadata.

**Usage**

```
initialise(config, script)
```

**Arguments**

config	a string specifying the location of the working config file in the data store
script	a string specifying the location of the submission script in the data store

**Value**

Returns an object of class `fdp`, R6 containing metadata required by the Data Pipeline API

---

is_queryable	<i>Check whether fields are queryable</i>
--------------	-------------------------------------------

---

**Description**

Check whether fields are queryable

**Usage**

```
is_queryable(table, query, method, endpoint)
```

**Arguments**

table	a string specifying the name of the table
query	a list containing the query
method	a string specifying the method, c("GET", "POST")
endpoint	endpoint

**Value**

Returns TRUE if the entry is queryable and FALSE if it isn't

---

link_read	<i>Link path to external format data</i>
-----------	------------------------------------------

---

**Description**

Link path to external format data

**Usage**

```
link_read(handle, data_product)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string representing an external object in the config.yaml file

**Value**

Returns a string specifying the location of the data product to be read

---

link_write	<i>Link path for external format data</i>
------------	-------------------------------------------

---

**Description**

Link path for external format data

**Usage**

```
link_write(handle, data_product)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string representing an external object in the config.yaml file

**Value**

Returns a string specifying the location in which the data product should be written

---

new_author	<i>Post entry to author table</i>
------------	-----------------------------------

---

**Description**

Upload information to the author table in the data registry

**Usage**

```
new_author(name, identifier, endpoint = "http://localhost:8000/api/")
```

**Arguments**

name	a string specifying the full name or organisation name of the author; note that at least one of name or identifier must be specified
identifier	(optional) a string specifying the full URL identifier ( <i>e.g.</i> ORCID or ROR ID) of the author
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_code_repo_release	<i>Post entry to code_repo_release table</i>
-----------------------	----------------------------------------------

---

**Description**

Upload information to the code\_repo\_release table in the data registry

**Usage**

```
new_code_repo_release(  
  name,  
  version,  
  object_url,  
  website,  
  endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

name	a string specifying the name of an official release of code
version	a string specifying the version release (conforming with semantic versioning syntax)
object_url	a string specifying the URL of an object
website	(optional) a string specifying the URL of the website for this code release
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_code_run	<i>Post entry to code_run table</i>
--------------	-------------------------------------

---

**Description**

Upload information to the code\_run table in the data registry

**Usage**

```
new_code_run(
  run_date,
  description,
  code_repo_url,
  model_config_url,
  submission_script_url,
  inputs_urls = list(),
  outputs_urls = list(),
  endpoint = "http://localhost:8000/api/"
)
```

**Arguments**

run_date	the date-time of the code_run <i>e.g.</i> Sys.time() or "2010-07-11 12:15:00 BST"
description	(optional) a string containing a free text description of the code_run
code_repo_url	(optional) a string specifying the URL of an object associated with the code_repo_release that was run
model_config_url	(optional) a string specifying the URL of an object associated with the working config file used for the code_run

submission_script_url	(optional) a string specifying the URL of an object associated with the submission script used for the code_run
inputs_urls	a list of object_component URLs referencing code_run inputs
outputs_urls	a list of object_component URLs referencing code_run outputs
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_data_product	<i>Post entry to data_product table</i>
------------------	-----------------------------------------

---

**Description**

Upload information to the data\_product table in the data registry

**Usage**

```
new_data_product(
  name,
  version,
  object_url,
  namespace_url,
  endpoint = "http://localhost:8000/api/"
)
```

**Arguments**

name	a string specifying the name of the data_product
version	a string specifying the version identifier of the data_product (must conform to semantic versioning syntax)
object_url	a string specifying the URL of the entry in the object table
namespace_url	a string specifying the URL of the entry in the namespace table
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new\_external\_object     *Post entry to external\_object table*

---

### Description

Upload information to the external\_object table in the data registry

### Usage

```
new_external_object(
  doi_or_unique_name,
  primary_not_supplement = TRUE,
  release_date,
  title,
  description,
  data_product_url,
  original_store_url,
  endpoint = "http://localhost:8000/api/"
)
```

### Arguments

`doi_or_unique_name`     a string specifying the DOI or name of the external\_object

`primary_not_supplement`     (optional) a boolean flag to indicate whether the external object is the primary source (TRUE) or not (FALSE)

`release_date`     the date-time that the external\_object was released *e.g.* `Sys.time()` or "2010-07-11 12:15:00 BST"

`title`     a string specifying the title of the external\_object

`description`     (optional) a string containing a free text description of the external\_object

`data_product_url`     a string specifying the URL of an entry in the data\_product table

`original_store_url`     (optional) a string specifying the URL of a an entry in the storage\_location table that references the original location of an external\_object

`endpoint`     a string specifying the registry endpoint

### See Also

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_file_type	<i>Post entry to file_type table</i>
---------------	--------------------------------------

---

**Description**

Upload information to the file\_type table in the data registry

**Usage**

```
new_file_type(name, extension, endpoint = "http://localhost:8000/api/")
```

**Arguments**

name	a string specifying the name of the file type
extension	a string specifying the filename extension
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_issue	<i>Post entry to issue table</i>
-----------	----------------------------------

---

**Description**

Upload information to the issue table in the data registry

**Usage**

```
new_issue(  
  severity,  
  description,  
  component_issues,  
  endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

severity	an integer specifying the severity of the issue
description	a string containing a free text description of the issue
component_issues	a list of object_component URLs with which the issue is associated; this can be an empty list
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_keyword	<i>Post entry to keyword table</i>
-------------	------------------------------------

---

**Description**

Upload information to the keyword table in the data registry

**Usage**

```
new_keyword(
  object_url,
  keyphrase,
  identifier,
  endpoint = "http://localhost:8000/api/"
)
```

**Arguments**

object_url	a string specifying the URL of an object
keyphrase	a string a string containing a free text key phrase
identifier	(optional) a string specifying the URL of ontology annotation to associate with this keyword
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)



---

new_licence	<i>Post entry to licence table</i>
-------------	------------------------------------

---

**Description**

Upload information to the licence table in the data registry

**Usage**

```
new_licence(object_url, licence_info, endpoint = "http://localhost:8000/api/")
```

**Arguments**

object_url	a string specifying the URL of an object
licence_info	a free text string containing information about the licence
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_namespace	<i>Post entry to namespace table</i>
---------------	--------------------------------------

---

**Description**

Upload information to the namespace table in the data registry

**Usage**

```
new_namespace(  
  name,  
  full_name,  
  website,  
  endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

name	a string specifying the name of the namespace
full_name	(optional) a string specifying the full name of the namespace
website	(optional) a string specifying the website URL associated with the namespace
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_object	<i>Post entry to object table</i>
------------	-----------------------------------

---

**Description**

Upload information to the object table in the data registry

**Usage**

```
new_object(
  description,
  storage_location_url,
  authors_url,
  file_type_url,
  endpoint = "http://localhost:8000/api/"
)
```

**Arguments**

`description` (optional) a string containing a free text description of the object

`storage_location_url` (optional) a string specifying the URL of an entry in the `storage_location` table

`authors_url` (optional) a list of author URLs associated with this object

`file_type_url` (optional) a string specifying the URL of an entry in the `file_type` table

`endpoint` a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new\_object\_component *Post entry to object\_component table*

---

### Description

Upload information to the object\_component table in the data registry

### Usage

```
new_object_component(  
    object_url,  
    name,  
    description,  
    whole_object = FALSE,  
    issues_urls,  
    endpoint = "http://localhost:8000/api/"  
)
```

### Arguments

object_url	a string specifying the URL of an existing object
name	a string specifying the name of the object_component, unique in the context of object_component and its object reference
description	(optional) a string containing a free text description of the object_component
whole_object	a boolean flag specifying whether or not this object_component refers to the whole object or not - default is FALSE
issues_urls	(optional) a list of issues URLs to associate with this object
endpoint	a string specifying the registry endpoint

Note that the object\_component table contains issues as an additional optional field. This is not included here. Instead use attach\_issue() and associated functionality to attach issues to objects and object components.

### See Also

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

new\_quality\_controlled

*Post entry to quality\_controlled table*

---

### **Description**

Upload information to the quality\_controlled table in the data registry

### **Usage**

```
new_quality_controlled(object_url, endpoint = "http://localhost:8000/api/")
```

### **Arguments**

object\_url      a string specifying the URL of an object  
endpoint        a string specifying the registry endpoint

### **See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new\_storage\_location    *Post entry to storage\_location table*

---

### **Description**

Upload information to the storage\_location table in the data registry

### **Usage**

```
new_storage_location(  
  path,  
  hash,  
  public,  
  storage_root_url,  
  endpoint = "http://localhost:8000/api/"  
)
```

**Arguments**

path	a string specifying the path from the storage_root URI to the item location, which when appended to storage_root URI produces a complete URL
hash	a string specifying the SHA1 hash of a file stored in storage_location
public	a boolean indicating whether the storage_location is public or not (default is TRUE)
storage_root_url	a string specifying the URL of an entry in the storage_root table
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_root\(\)](#), [new\\_user\\_author\(\)](#)

---

new_storage_root	<i>Post entry to storage_root table</i>
------------------	-----------------------------------------

---

**Description**

Upload information to the storage\_root table in the data registry

**Usage**

```
new_storage_root(root, local, endpoint = "http://localhost:8000/api/")
```

**Arguments**

root	a string specifying the URI of a storage_location, which when prepended to a storage_location produces a complete URI to a file
local	(optional) a boolean indicating whether the storage_root is local or not
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_user\\_author\(\)](#)

---

new_user_author	<i>Post entry to user_author table</i>
-----------------	----------------------------------------

---

**Description**

Upload information to the user\_author table in the data registry

**Usage**

```
new_user_author(user_url, author_url, endpoint = "http://localhost:8000/api/")
```

**Arguments**

user_url	a string specifying the URL of an existing user
author_url	a string specifying the URL of an existing author
endpoint	a string specifying the registry endpoint

**See Also**

Other new functions: [new\\_author\(\)](#), [new\\_code\\_repo\\_release\(\)](#), [new\\_code\\_run\(\)](#), [new\\_data\\_product\(\)](#), [new\\_external\\_object\(\)](#), [new\\_file\\_type\(\)](#), [new\\_issue\(\)](#), [new\\_keyword\(\)](#), [new\\_licence\(\)](#), [new\\_namespace\(\)](#), [new\\_object\\_component\(\)](#), [new\\_object\(\)](#), [new\\_quality\\_controlled\(\)](#), [new\\_storage\\_location\(\)](#), [new\\_storage\\_root\(\)](#)

---

paper_exists	<i>Check whether paper exists</i>
--------------	-----------------------------------

---

**Description**

Check whether paper is in the data registry

**Usage**

```
paper_exists(doi)
```

**Arguments**

doi	doi
-----	-----

---

raise_issue	<i>raise_issue</i>
-------------	--------------------

---

**Description**

raise\_issue

**Usage**

```
raise_issue(
  index,
  handle,
  component = NA,
  data_product,
  issue,
  severity,
  whole_object = FALSE
)
```

**Arguments**

index	index returned from link_*( ), read_(), or write()
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
component	a string specifying the component name
data_product	a string specifying the data product name
issue	a string specifying the issue
severity	a numeric value specifying the severity
whole_object	a boolean flag specifying whether or not to reference the whole_object

---

raise_issue_config	<i>Raise issue with config file</i>
--------------------	-------------------------------------

---

**Description**

Raise issue with config file

**Usage**

```
raise_issue_config(handle, issue, severity)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity

---

raise\_issue\_repo      *Raise issue with remote repository*

---

**Description**

Raise issue with remote repository

**Usage**

```
raise_issue_repo(handle, issue, severity)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity

---

raise\_issue\_script      *Raise issue with submission script*

---

**Description**

Raise issue with submission script

**Usage**

```
raise_issue_script(handle, issue, severity)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
issue	a string specifying the issue
severity	a numeric value specifying the severity



---

random_hash	<i>random_hash</i>
-------------	--------------------

---

**Description**

Generates a random hash

**Usage**

random\_hash()

---

read_array	<i>Read array component from HDF5 file</i>
------------	--------------------------------------------

---

**Description**

Function to read array type data from hdf5 file.

**Usage**

read\_array(handle, data\_product, component)

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

**Value**

Returns an array with attached Dimension\_i\_title, Dimension\_i\_units, Dimension\_i\_values, and units attributes, if available

---

read_distribution	<i>Read distribution component from TOML file</i>
-------------------	---------------------------------------------------

---

**Description**

Function to read distribution type data from toml file.

**Usage**

```
read_distribution(handle, data_product, component)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

---

read_estimate	<i>Read estimate component from TOML file</i>
---------------	-----------------------------------------------

---

**Description**

Function to read point-estimate type data from toml file.

**Usage**

```
read_estimate(handle, data_product, component)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

---

read_table	<i>Read table component from HDF5 file</i>
------------	--------------------------------------------

---

**Description**

Function to read table type data from hdf5 file.

**Usage**

```
read_table(handle, data_product, component)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying a data product
component	a string specifying a data product component

**Value**

Returns a data.frame with attached column\_units attributes, if available

---

register_issue_dataproduct	<i>register_issue_dataproduct</i>
----------------------------	-----------------------------------

---

**Description**

register\_issue\_dataproduct

**Usage**

```
register_issue_dataproduct(handle, this_issue)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
this_issue	this_issue

---

`register_issue_script` *register\_issue\_script*

---

**Description**

`register_issue_script`

**Usage**

`register_issue_script(handle, this_issue, type)`

**Arguments**

<code>handle</code>	an object of class <code>fdp</code> , <code>R6</code> containing metadata required by the Data Pipeline API
<code>this_issue</code>	<code>this_issue</code>
<code>type</code>	<code>type</code>

---

`remove_empty_parents` *remove\_empty\_parents*

---

**Description**

`remove_empty_parents`

**Usage**

`remove_empty_parents(path, root)`

**Arguments**

<code>path</code>	<code>path</code>
<code>root</code>	<code>root</code>

---

resolve_read	<i>resolve_read</i>
--------------	---------------------

---

**Description**

resolve\_read

**Usage**

```
resolve_read(handle, data_product, component = NA)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying the name of data product component

---

resolve_version	<i>resolve_version</i>
-----------------	------------------------

---

**Description**

resolve\_version

**Usage**

```
resolve_version(version, data_product, namespace_id)
```

**Arguments**

version	version number
data_product	data_product
namespace_id	namespace_id

---

resolve_write	<i>resolve_data_product</i>
---------------	-----------------------------

---

**Description**

resolve\_data\_product

**Usage**

```
resolve_write(handle, data_product, file_type)
```

**Arguments**

handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
file_type	(optional) a string specifying the file type; when missing, file_type will be read from the config file

---

validate_fields	<i>Validate fields</i>
-----------------	------------------------

---

**Description**

Function to validate fields in post data

**Usage**

```
validate_fields(table, data, endpoint)
```

**Arguments**

table	table name as character
data	data as a named list
endpoint	endpoint

**Value**

Returns

---

write_array	<i>Write array component to HDF5 file</i>
-------------	-------------------------------------------

---

### Description

Function to populate hdf5 file with array type data.

### Usage

```
write_array(
  array,
  handle,
  data_product,
  component,
  description,
  dimension_names,
  dimension_values,
  dimension_units,
  units
)
```

### Arguments

array	an array containing the data
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the hdf5 file
description	a string describing the data product component
dimension_names	a list where each element is a vector containing the labels associated with a particular dimension (e.g. element 1 corresponds to dimension 1, which corresponds to row names) and the name of each element describes the contents of each dimension (e.g. age classes).
dimension_values	(optional) a list of values corresponding to each dimension (e.g. list element 2 corresponds to columns)
dimension_units	(optional) a list of units corresponding to each dimension (e.g. list element 2 corresponds to columns)
units	(optional) a string specifying the units of the data as a whole

### Value

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

**See Also**

Other write functions: [write\\_distribution\(\)](#), [write\\_estimate\(\)](#), [write\\_table\(\)](#)

---

write_distribution	<i>Write distribution component to TOML file</i>
--------------------	--------------------------------------------------

---

**Description**

Write distribution component to TOML file

**Usage**

```
write_distribution(  
  distribution,  
  parameters,  
  handle,  
  data_product,  
  component,  
  description  
)
```

**Arguments**

distribution	a string specifying the name of the distribution
parameters	a list specifying the distribution parameters
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the toml file
description	a string describing the data product component

**Value**

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

**See Also**

Other write functions: [write\\_array\(\)](#), [write\\_estimate\(\)](#), [write\\_table\(\)](#)



---

write_estimate	<i>Write estimate component to TOML file</i>
----------------	----------------------------------------------

---

**Description**

Function to populate toml file with point-estimate type data. If a file already exists at the specified location, an additional component will be added.

**Usage**

```
write_estimate(value, handle, data_product, component, description)
```

**Arguments**

value	an object of class numeric
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the toml file
description	a string describing the data product component

**Value**

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

**See Also**

Other write functions: [write\\_array\(\)](#), [write\\_distribution\(\)](#), [write\\_table\(\)](#)

---

write_table	<i>Write table component to HDF5 file</i>
-------------	-------------------------------------------

---

**Description**

Function to populate hdf5 file with array type data.

**Usage**

```
write_table(  
  df,  
  handle,  
  data_product,  
  component,  
  description,  
  row_names,  
  column_units  
)
```

**Arguments**

df	an dataframe containing the data
handle	an object of class fdp, R6 containing metadata required by the Data Pipeline API
data_product	a string specifying the name of the data product
component	a string specifying a location within the hdf5 file,
description	a string describing the data product component
row_names	(optional) a vector of rownames
column_units	(optional) a vector comprising column units

**Value**

Returns a handle index associated with the just written component, which can be used to raise an issue if necessary

**See Also**

Other write functions: [write\\_array\(\)](#), [write\\_distribution\(\)](#), [write\\_estimate\(\)](#)

# Index

- \* **create functions**
  - create\_version\_number, 14
- \* **download functions**
  - download\_from\_database, 14
  - download\_from\_url, 15
- \* **get functions**
  - get\_components, 25
  - get\_entry, 27
  - get\_existing, 27
  - get\_file\_hash, 29
  - get\_github\_hash, 29
- \* **new functions**
  - new\_author, 35
  - new\_code\_repo\_release, 35
  - new\_code\_run, 36
  - new\_data\_product, 37
  - new\_external\_object, 38
  - new\_file\_type, 39
  - new\_issue, 39
  - new\_keyword, 40
  - new\_licence, 41
  - new\_namespace, 41
  - new\_object, 42
  - new\_object\_component, 43
  - new\_quality\_controlled, 44
  - new\_storage\_location, 44
  - new\_storage\_root, 45
  - new\_user\_author, 46
- \* **write functions**
  - write\_array, 55
  - write\_distribution, 56
  - write\_estimate, 57
  - write\_table, 57
- add\_read, 4
- add\_write, 5
- check\_config, 6
- check\_dataproduct\_exists, 6
- check\_datetime, 7
- check\_exists, 8
- check\_field, 8
- check\_fields, 9
- check\_handle, 9
- check\_integer, 10
- check\_local\_repo, 10
- check\_string, 11
- check\_table\_exists, 11
- check\_yaml\_write, 12
- clean\_query, 12
- create\_config, 13
- create\_index, 13
- create\_version\_number, 14
- download\_from\_database, 14, 15
- download\_from\_url, 15, 15
- extract\_id, 16
- fair\_init, 16
- fair\_run, 17
- fdp (fdp-class), 17
- fdp-class, 17
- fdp\_resolve\_read, 22
- fdp\_resolve\_write, 22
- finalise, 23
- find\_read\_match, 24
- find\_write\_match, 24
- findme, 23
- get\_author\_url, 25
- get\_components, 25, 27–29
- get\_dataproduct, 26
- get\_entity, 26
- get\_entry, 25, 27, 28, 29
- get\_existing, 25, 27, 27, 29
- get\_fields, 28
- get\_file\_hash, 25, 27–29, 29
- get\_github\_hash, 25, 27–29, 29
- get\_id, 30

get\_index, 30  
get\_max\_version, 31  
get\_storage\_location, 31  
get\_token, 31  
get\_url, 32

increment\_filename, 32  
initialise, 33  
is\_queryable, 33

link\_read, 34  
link\_write, 34

new\_author, 35, 36–46  
new\_code\_repo\_release, 35, 35, 37–46  
new\_code\_run, 35, 36, 36, 37–46  
new\_data\_product, 35–37, 37, 38–46  
new\_external\_object, 35–37, 38, 39–46  
new\_file\_type, 35–38, 39, 40–46  
new\_issue, 35–39, 39, 40–46  
new\_keyword, 35–40, 40, 41–46  
new\_licence, 35–40, 41, 42–46  
new\_namespace, 35–41, 41, 42–46  
new\_object, 35–42, 42, 43–46  
new\_object\_component, 35–42, 43, 44–46  
new\_quality\_controlled, 35–43, 44, 45, 46  
new\_storage\_location, 35–44, 44, 45, 46  
new\_storage\_root, 35–45, 45, 46  
new\_user\_author, 35–45, 46

paper\_exists, 46

raise\_issue, 47  
raise\_issue\_config, 47  
raise\_issue\_repo, 48  
raise\_issue\_script, 48  
random\_hash, 49  
rDataPipeline (rDataPipeline-package), 3  
rDataPipeline-package, 3  
read\_array, 49  
read\_distribution, 50  
read\_estimate, 50  
read\_table, 51  
register\_issue\_dataproduct, 51  
register\_issue\_script, 52  
remove\_empty\_parents, 52  
resolve\_read, 53  
resolve\_version, 53  
resolve\_write, 54  
validate\_fields, 54  
write\_array, 55, 56–58  
write\_distribution, 56, 56, 57, 58  
write\_estimate, 56, 57, 58  
write\_table, 56, 57, 57