Package ‘rirods’

March 15, 2024

Title R Client for 'iRODS'

Version 0.2.0

Description The open sourced data management software 'Integrated Rule-Oriented Data System' ('iRODS') offers solutions for the whole data life cycle (<https://irods.org/>). The loosely constructed and highly configurable architecture of 'iRODS' frees the user from strict formatting constraints and single-vendor solutions. This package provides an interface to the 'iRODS' HTTP API, allowing you to manage your data and metadata in 'iRODS' with R. Storage of annotated files and R objects in 'iRODS' ensures findability, accessibility, interoperability, and reusability of data.

License MIT + file LICENSE

URL https://github.com/irods/irods_client_library_rirods,
https://rirods.irods4r.org

BugReports https://github.com/irods/irods_client_library_rirods/issues

Depends R (>= 4.1)

Imports curl, httr2 (>= 0.2.2), jsonlite, rappdirs, stats, testthat (>= 3.0.0), utils, withr

Suggests httptest2, kableExtra, knitr, purrr, readr, rmarkdown, spelling

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

Language en-US

RoxygenNote 7.3.1

SystemRequirements iRODS C++ HTTP API 0.2.0 (<https://github.com/irods/irods_client_http_api>)

NeedsCompilation no

Author Martin Schobben [aut, cre, cph]
(<https://orcid.org/0000-0001-8560-0037>),
Mariana Montes [aut],
Terrell Russell [ctb],
Christine Staiger [ctb],
Ton Smeele [ctb],
Alan King [ctb]

Maintainer Martin Schobben <schobbenmartin@gmail.com>

Repository CRAN

Date/Publication 2024-03-15 18:30:02 UTC

R topics documented:

`as.data.frame.irods_df`

Coerce to a Data Frame

Description

Coerce iRODS Zone information class to `data.frame()`.

Usage

```
## S3 method for class 'irods_df'
as.data.frame(x, ...)
```

Arguments

x	`irods_df` class object.
...	Currently not implemented

Value

Returns a `data.frame`. Note, that the columns of metadata consists of a list of data frames, and status_information and permission_information consist of data frames.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# some data
foo <- data.frame(x = c(1, 8, 9), y = c("x", "y", "z"))

# store data in iRODS
isaveRDS(foo, "foo.rds")

# add some metadata
imeta(
  "foo.rds",
  operations =
    data.frame(operation = "add", attribute = "foo", value = "bar",
      units = "baz")
)

# iRODS Zone with metadata
irods_zone <- ils(metadata = TRUE)

# check class
class(irods_zone)

# coerce into `data.frame` and extract metadata of "foo.rds"
irods_zone <- as.data.frame(irods_zone)
irods_zone[basename(irods_zone$logical_path) == "foo.rds", "metadata"]

# delete object
irm("foo.rds", force = TRUE)
```

`create_irods` *Generate IRODS Configuration File*

Description

This will create an iRODS configuration file containing information about the iRODS server. Once the file has been created, future sessions connect again with the same iRODS server without further intervention.

Usage

```
create_irods(host, zone_path = character(1), overwrite = FALSE)
```

Arguments

`host`	URL of host.
`zone_path`	Deprecated
`overwrite`	Overwrite existing iRODS configuration file. Defaults to `FALSE`.

Details

The configuration file is located in the user-specific configuration directory. This destination is set with R_USER_CONFIG_DIR if set. Otherwise, it follows platform conventions (see also `rappdirs::user_config_dir()`).

Value

Invisibly, the path to the iRODS configuration file.

`iadmin` *The Administration Interface to iRODS*

Description

Note that this function can only be used with admin rights.

Usage

```
iadmin(
  name,
  password = character(1),
  action = c("create_user", "set_password", "remove_user"),
  role = c("rodsuser", "groupadmin", "rodsadmin"),
  verbose = FALSE
)
```

Arguments

`name`	Name of user to be added.
`password`	Password to be added.
`action`	The action: `"create_user"`, `"remove_user"`, or `"set_password"`.
`role`	Role of user: `"rodsuser"`, `"groupadmin"`, and `"groupadmin"`.
`verbose`	Show information about the http request and response. Defaults to `FALSE`.

Value

Invisible http status.

Examples

```
is_irods_demo_running()

# demonstration server (requires Bash, Docker and Docker-compose)
# use_irods_demo()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authentication
iauth("rods", "rods")

# add user
iadmin("Alice", action = "create_user", role = "rodsuser")

# add user password
iadmin("Alice", "pass", action = "set_password",  role = "rodsuser")

# delete user
iadmin("Alice", action = "remove_user", role = "rodsuser")
```

`iauth` *Authentication Service for an iRODS Zone*

Description

Provides an authentication service for an iRODS zone. Using the function without arguments results in a prompt asking for the user name and password thereby preventing hard-coding of sensitive information in scripts.

Usage

```
iauth(user, password = NULL, role = "rodsuser")
```

Arguments

`user`	iRODS user name (prompts user for user name if not supplied).
`password`	iRODS password (prompts user for password if not supplied).
`role`	iRODS role of user (defaults to `"rodsuser"`).

Value

Invisibly `NULL`.

Examples

```
is_irods_demo_running()

# demonstration server (requires Bash, Docker and Docker-compose)
# use_irods_demo()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")
```

`icd` *Get or Set Current Working Directory in iRODS*

Description

`ipwd()` and `icd()` are the iRODS equivalents of `getwd()` and `setwd()` respectively. For example, whereas `getwd()` returns the current working directory in the local system, `ipwd()` returns the current working directory in iRODS.

Usage

```
icd(dir)

ipwd()
```

Arguments

`dir`	Collection to set as working directory.

Value

Invisibly the current directory before the change (same convention as `setwd()`).

See Also

`setwd()` and `getwd()` for R equivalents, `ils()` for listing collections and objects in iRODS.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods", "rodsadmin")

# default dir
icd(".")
ipwd()

# relative paths work as well
icd("/tempZone/home")
ipwd()

# go back on level lower
icd("..")
ipwd()

# absolute paths work as well
icd("/tempZone/home/rods")
ipwd()

# back home
icd("/tempZone/home")
```

`iget` *Retrieve File or Object from iRODS*

Description

Transfer a file from iRODS to the local storage with `iget()` or read an R object from an RDS file in iRODS with `ireadRDS()` (see `readRDS()`).

Usage

```
iget(
  logical_path,
  local_path,
  offset = 0,
  count = 0,
  verbose = FALSE,
  overwrite = FALSE,
  ticket = NULL
)

ireadRDS(logical_path, offset = 0, count = 0, verbose = FALSE, ticket = NULL)
```

Arguments

`logical_path`	Source path in iRODS.
`local_path`	Destination path in local storage. By default, the basename of the logical path; the file will be stored in the current directory (see `getwd()`).
`offset`	Offset in bytes into the data object. Deprecated.
`count`	Maximum number of bytes to write. Deprecated.
`verbose`	Whether information should be printed about the HTTP request and response.
`overwrite`	Whether the local file should be overwritten if it exists. Defaults to `FALSE`.
`ticket`	A valid iRODS ticket string. Defaults to `NULL`.

Value

The R object in case of `ireadRDS()`, invisibly `NULL` in case of `iget()`.

The R object in case of `ireadRDS()`, invisibly `NULL` in case of `iget()`.

See Also

`iput()` for sending files, `isaveRDS()` for sending R objects to iRODS, `saveRDS()` for an R equivalent.

Transfer a file from iRODS to the local storage with `iget()` or read an R object from an RDS file in iRODS with `ireadRDS()` (see `readRDS()`).

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# save the iris dataset as csv and send the file to iRODS
write.csv(iris, "iris.csv")
iput("iris.csv", "iris.csv")

# bring the file back with a different name
iget("iris.csv", "newer_iris.csv")
```

```
file.exists("newer_iris.csv") # check that it has been transferred

# send an R object to iRODS in RDS format
isaveRDS(iris, "irids_in_rds.rds")

# read it back
iris_again <- ireadRDS("irids_in_rds.rds")
iris_again

# delete objects in iRODS
irm("irids_in_rds.rds", force = TRUE)
irm("iris.csv", force = TRUE)
```

`ils` *List iRODS Data Objects and Collections*

Description

List the contents of a collection, optionally with stat, metadata, and/or access control information for each element in the collection.

Usage

```
ils(
  logical_path = ".",
  stat = FALSE,
  permissions = FALSE,
  metadata = FALSE,
  offset = numeric(1),
  limit = find_irods_file("max_number_of_rows_per_catalog_query"),
  recurse = FALSE,
  ticket = NULL,
  message = TRUE,
  verbose = FALSE
)
```

Arguments

`logical_path`	Path to the collection whose contents are to be listed. By default this is the current working collection (see `ipwd()`).
`stat`	Whether stat information should be included. Defaults to `FALSE`.
`permissions`	Whether access control information should be included. Defaults to `FALSE`.
`metadata`	Whether metadata information should be included. Defaults to `FALSE`.
`offset`	Number of records to skip for pagination. Deprecated.

`limit`	Number of records to show per page.
`recurse`	Recursively list. Defaults to `FALSE`.
`ticket`	A valid iRODS ticket string. Defaults to `NULL`.
`message`	Show message when empty collection. Default to `FALSE`.
`verbose`	Whether information should be printed about the HTTP request and response. Defaults to `FALSE`.

Value

Dataframe with logical paths and, if requested, additional information.

See Also

`ipwd()` for finding the working collection, `ipwd()` for setting the working collection, and `list.files()` for an R equivalent.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# list home directory
ils()

# make collection
imkdir("some_collection")

# list a different directory
ils("/tempZone/home/rods/some_collection")

# show metadata
ils(metadata = TRUE)

# delete `some_collection`
irm("some_collection", force = TRUE, recursive = TRUE)
```

imeta *Add or Remove Metadata*

Description

In iRODS, metadata is stored as attribute-value-units triples (AVUs), consisting of an attribute name, an attribute value and an optional unit. This function allows to chain several operations ('add' or 'remove') linked to specific AVUs. Read more about metadata by looking at the iCommands equivalent `imeta` in the iRODS Docs.

Usage

```
imeta(
  logical_path,
  entity_type = c("data_object", "collection", "user"),
  operations = list(),
  admin = FALSE,
  verbose = FALSE
)
```

Arguments

`logical_path`	Path to the data object or collection (or name of the user).
`entity_type`	Type of item to add metadata to or remove it from. Options are 'data_object', 'collection' and 'user'. Deprecated.
`operations`	List of named lists or data.frame representing operations. The valid components of each of these lists or vectors are: • `operation`, with values 'add' or 'remove', depending on whether the AVU should be added to or removed from the metadata of the item (required). • `attribute`, with the name of the AVU (required). • `value`, with the value of the AVU (required). • `units`, with the unit of the AVU (optional).
`admin`	Whether to grant admin rights. Defaults to `FALSE`.
`verbose`	Whether information should be printed about the HTTP request and response. Defaults to `FALSE`.

Value

Invisibly, the HTTP response.

References

https://docs.irods.org/master/icommands/metadata/

See Also

iquery()

Examples

```
is_irods_demo_running()

# demonstration server (requires Bash, Docker and Docker-compose)
# use_irods_demo()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authentication
iauth("rods", "rods")

# some data
foo <- data.frame(x = c(1, 8, 9), y = c("x", "y", "z"))

# store
isaveRDS(foo, "foo.rds")

# check if file is stored
ils()

# add some metadata
imeta(
  "foo.rds",
   operations =
    list(
     list(operation = "add", attribute = "foo", value = "bar", units = "baz")
   )
)

# `operations` can contain multiple tags supplied as a `data.frame`
imeta(
  "foo.rds",
  operations = data.frame(
    operation = c("add", "add"),
    attribute = c("foo2", "foo3"),
    value = c("bar2", "bar3"),
    units = c("baz2", "baz3")
   )
 )

# or again as a list of lists
imeta(
  "foo.rds",
  operations = list(
    list(operation = "add", attribute = "foo4", value = "bar4", units = "baz4"),
    list(operation = "add", attribute = "foo5", value = "bar5", units = "baz5")
  )
)

# list of lists are useful as AVUs don't have to contain units
imeta(
  "foo.rds",
  operations = list(
    list(operation = "add", attribute = "foo6", value = "bar6"),
    list(operation = "add", attribute = "foo7", value = "bar7", units = "baz7")
  )
)

# check if file is stored with associated metadata
ils(metadata = TRUE)

# delete object
irm("foo.rds", force = TRUE)
```

imkdir — *Create a New Collection in iRODS*

Description

This is the equivalent to `dir.create()`, but creating a collection in iRODS instead of a local directory.

Usage

```
imkdir(
  logical_path,
  create_parent_collections = FALSE,
  overwrite = FALSE,
  verbose = FALSE
)
```

Arguments

`logical_path`	Path to the collection to create, relative to the current working directory (see `ipwd()`).
`create_parent_collections`	Whether parent collections should be created when necessary. Defaults to `FALSE`.
`overwrite`	Whether the existing collection should be overwritten if it exists. Defaults to `FALSE`.
`verbose`	Whether information about the HTTP request and response should be printed. Defaults to `FALSE`.

Value

Invisibly the HTTP request.

See Also

`irm()` for removing collections, `dir.create()` for an R equivalent.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authentication
iauth("rods", "rods")

# list all object and collection in the current collection of iRODS
ils()

# create a new collection
imkdir("new_collection")

# check if it is there
ils()

# and move to the new directory
icd("new_collection")

# remove collection
icd("..")
irm("new_collection", force = TRUE, recursive = TRUE)
```

`iput` *Save Files and Objects in iRODS*

Description

Store an object or file into iRODS. `iput()` should be used to transfer a file from the local storage to iRODS; `isaveRDS()` saves an R object from the current environment in iRODS in RDS format (see `saveRDS()`).

Usage

```
iput(
  local_path,
  logical_path,
  offset = 0,
  count = 0,
  truncate = TRUE,
  verbose = FALSE,
  overwrite = FALSE,
  ticket = NULL
)

isaveRDS(
  x,
  logical_path,
  offset = 0,
  count = 0,
  truncate = TRUE,
  verbose = FALSE,
  overwrite = FALSE,
  ticket = NULL
)
```

Arguments

`local_path`	Local path of file to be sent to iRODS.
`logical_path`	Destination path in iRODS.
`offset`	Offset in bytes into the data object. Deprecated.
`count`	Maximum number of bytes to write. Deprecated.
`truncate`	Whether to truncate the object when opening it. Deprecated.
`verbose`	Whether to print information about the HTTP request and response. Defaults to `FALSE`.
`overwrite`	Whether the file in iRODS should be overwritten if it exists. Defaults to `FALSE`.
`ticket`	A valid iRODS ticket string. Defaults to `NULL`.
`x`	R object to save in iRODS.

Value

(Invisibly) the HTTP response.

See Also

`iget()` for obtaining files, `ireadRDS()` for obtaining R objects from iRODS, `readRDS()` for an R equivalent.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# save the iris dataset as csv and send the file to iRODS
write.csv(iris, "iris.csv")
iput("iris.csv", "iris.csv")

# save with a different name
iput("iris.csv", "iris_in_irods.csv")
ils()

# send an R object to iRODS in RDS format
isaveRDS(iris, "iris_in_rds.rds")

# delete objects in iRODS
irm("iris_in_irods.csv", force = TRUE)
irm("iris_in_rds.rds", force = TRUE)
irm("iris.csv", force = TRUE)
```

---

iquery *Query Data Objects and Collections in iRODS*

Description

Use SQL-like expressions to query data objects and collections based on different properties. Read more about queries by looking at the iCommands equivalent `iquest` in the iRODS Docs.

Usage

```
iquery(
  query,
  limit = 100,
  offset = 0,
  type = c("general", "specific"),
  case_sensitive = TRUE,
  distinct = TRUE,
  parser = c("genquery1", "genquery2"),
  sql_only = FALSE,
  verbose = FALSE
)
```

Arguments

`query`	GeneralQuery for searching the iCAT database.
`limit`	Maximum number of rows to return.
`offset`	Number of rows to skip for paging. Deprecated.
`type`	Type of query: 'general' (the default) or 'specific'.
`case_sensitive`	Whether the string matching in the query is case sensitive. Defaults to `TRUE`.
`distinct`	Whether only distinct rows should be listed. Defaults to `TRUE`.
`parser`	Which parser to use: genquery1 or genquery2. Defaults to genquery1.
`sql_only`	Whether to dry-run query and return SQL syntax query as return. Defaults to `FALSE`. Needs Genquery2.
`verbose`	Whether information should be printed about the HTTP request and response.

Value

A dataframe with one row per result and one column per requested attribute, with "size" and "time" columns parsed to the right type.

Invisibly, the HTTP response.

References

https://docs.irods.org/master/icommands/user/#iquest

Use SQL-like expressions to query data objects and collections based on different properties.

See Also

`imeta()`

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authentication
iauth("rods", "rods")

# some data
foo <- data.frame(x = c(1, 8, 9), y = c("x", "y", "z"))

# store
isaveRDS(foo, "foo.rds")

# add metadata
imeta(
  "foo.rds",
  operations =
    list(
      list(operation = "add", attribute = "bar", value = "baz")
  )
)

# search for objects by metadata
iquery("SELECT COLL_NAME, DATA_NAME WHERE META_DATA_ATTR_NAME LIKE 'bar%'")

# delete object
irm("foo.rds", force = TRUE)
```

irm — *Remove Data Objects or Collections in iRODS*

Description

This is the equivalent of `file.remove()`, but applied to an item inside iRODS.

Usage

```
irm(
  logical_path,
  force = TRUE,
  recursive = FALSE,
  catalog_only = FALSE,
  verbose = FALSE
)
```

Arguments

`logical_path`	Path to the data object or collection to remove.
`force`	Whether the data object or collection should be deleted permanently. If `FALSE`, it is sent to the trash collection. Defaults to `TRUE`.
`recursive`	If a collection is provided, whether its contents should also be removed. If a collection is not empty and `recursive` is `FALSE` , it cannot be deleted. Defaults to `FALSE`.
`catalog_only`	Whether to remove only the catalog entry. Defaults to `FALSE`.
`verbose`	Whether information should be printed about the HTTP request and response. Defaults to `FALSE`.

Value

Invisibly the HTTP call.

See Also

`imkdir()` for creating collections, `file.remove()` for an R equivalent.

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# some data
foo <- data.frame(x = c(1, 8, 9), y = c("x", "y", "z"))

# store
isaveRDS(foo, "foo.rds")

# check if file is stored
ils()

# delete object
irm("foo.rds", force = TRUE)

# check if file is deleted
ils()
```

`is_connected_irods` *Predicate for iRODS Connectivity*

Description

A predicate to check whether you are currently connected to an iRODS server.

Usage

```
is_connected_irods(...)
```

Arguments

`...`	Currently not implemented.

Value

Boolean whether or not a connection to iRODS exists.

Examples

```
is_connected_irods()
```

`is_irods_demo_running` *Predicate for iRODS Demonstration Service State*

Description

A predicate to check whether you are running iRODS docker demo containers.

Usage

```
is_irods_demo_running(...)
```

Arguments

`...`	Currently not implemented.

Value

Boolean whether or not connected to iRODS

Examples

```
is_irods_demo_running()
```

`print.irods_df` *Print Method for iRODS Data Frame Class.*

Description

Print Method for iRODS Data Frame Class.

Usage

```
## S3 method for class 'irods_df'
print(
  x,
  ...,
  digits = NULL,
  quote = FALSE,
  right = TRUE,
  row.names = FALSE,
  max = NULL,
  message = TRUE
)
```

Arguments

`x`	An object of class `irods_df`.
`...`	optional arguments to `print` methods.
`digits`	the minimum number of significant digits to be used: see `print.default`.
`quote`	logical, indicating whether or not entries should be printed with surrounding quotes.
`right`	logical, indicating whether or not strings should be right-aligned. The default is right-alignment.
`row.names`	logical (or character vector), indicating whether (or what) row names should be printed.
`max`	numeric or `NULL`, specifying the maximal number of entries to be printed. By default, when `NULL`, `getOption("max.print")` used.
`message`	Show message when empty collection. Default to `TRUE`.

Value

Invisibly return the class `irods_df` object.

See Also

`print.data.frame()`

Examples

```
is_irods_demo_running()

# connect project to server
create_irods("http://localhost:9001/irods-http-api/0.2.0")

# authenticate
iauth("rods", "rods")

# some data
foo <- data.frame(x = c(1, 8, 9), y = c("x", "y", "z"))

# store data in iRODS
isaveRDS(foo, "foo.rds")

# add some metadata
imeta(
  "foo.rds",
  operations =
   data.frame(operation = "add", attribute = "foo", value = "bar",
     units = "baz")
)

# iRODS Zone with metadata
irods_zone <- ils(metadata = TRUE)
```

```
# print (default no row.names)
print(irods_zone)

# with row.names
print(irods_zone, row.names = TRUE)

# delete object
irm("foo.rds", force = TRUE)
```

`use_irods_demo` *Run Docker iRODS Demonstration Service*

Description

Run an iRODS demonstration server with `use_irods_demo()` as a Docker container instance. The function `stop_irods_demo()` stops the containers.

Usage

```
use_irods_demo(
  user = character(),
  pass = character(),
  recreate = FALSE,
  verbose = TRUE
)

stop_irods_demo(verbose = TRUE)
```

Arguments

`user`	Character vector for user name (defaults to "rods" admin)
`pass`	Character vector for password (defaults to "rods" admin password)
`recreate`	Boolean to indicate whether to recreate (reboot) the iRODS demo server (defaults to `FALSE`). Recreating will destroy all content on the current instance.
`verbose`	Verbosity (defaults to `TRUE`).

Details

These functions are untested on Windows and macOS and require:

- `bash`
- `docker`

Value

Invisible

References

https://github.com/irods/irods_demo

Examples

```
if (interactive()) {

  # launch docker irods_demo containers (and possibly download images) with
  # default credentials
  use_irods_demo()

  # same but then with "alice" as user and "PASSword" as password
  use_irods_demo("alice", "PASSword")

  # stop containers
  stop_irods_demo()
}
```

Index