Package: rcdf (via r-universe)

May 30, 2026

Type Package

Title A Comprehensive Toolkit for Working with Encrypted Parquet Files

Version 0.1.6

Description Utilities for reading, writing, and managing RCDF files, including encryption and decryption support. It offers a flexible interface for handling data stored in encrypted Parquet format, along with metadata extraction, key management, and secure operations using AES and RSA encryptions.

Author Bhas Abdulsamad [aut, cre, cph] (<https://orcid.org/0009-0002-5891-8124>)

Maintainer Bhas Abdulsamad <aeabdulsamad@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports duckdb, haven, zip, utils (>= 4.0.0), openssl (>= 2.1.1), dplyr (>= 1.1.0), jsonlite (>= 1.8.0), DBI (>= 1.1.0), tools, stats

Suggests arrow, dbplyr (>= 2.4.0), openxlsx, RSQLite (>= 2.2.0), lifecycle, testthat, cli, devtools, knitr, rmarkdown, rlang (>= 1.0.2), tibble, withr, gt (>= 0.10.0)

Config/testthat/edition 3

RoxygenNote 7.3.3

VignetteBuilder knitr

Depends R (>= 4.1.0)

URL https://yng-me.github.io/rcdf/, https://github.com/yng-me/rcdf

BugReports https://github.com/yng-me/rcdf/issues

Roxygen list(markdown = TRUE)

Config/pak/sysreqs make libssl-dev libx11-dev xz-utils zlib1g-dev

Repository https://yng-me.r-universe.dev

Date/Publication 2026-05-30 17:19:45 UTC

RemoteUrl https://github.com/yng-me/rcdf

RemoteRef HEAD

RemoteSha 8324555fa64395078e34642ce859e58fdd927aef

Contents

`add_metadata` *Add metadata attributes to a data frame*

Description

Adds variable labels and value labels to a data frame based on a metadata dictionary. This is particularly useful for preparing datasets for use with packages like `haven` or for exporting to formats like SPSS or Stata.

Usage

```
add_metadata(data, metadata, ..., set_data_types = FALSE)
```

Arguments

`data`	A data frame containing the raw dataset.
`metadata`	A data frame that serves as a metadata dictionary. It must contain at least the columns: `variable_name`, `label`, and `type`. Optionally, it may include a `valueset` column for categorical variables, which should be a list column with data frames containing `value` and `label` columns.
`...`	Additional arguments (currently unused).
`set_data_types`	Logical; if `TRUE`, attempts to coerce column data types to match those implied by the metadata. (Note: currently not fully implemented.)

Details

The function first checks the structure of the `metadata` using an internal helper. Then, for each variable listed in `metadata`, it:

- Adds a label using the `label` attribute
- Converts values to labelled vectors using `haven::labelled()` if a `valueset` is provided

If value labels are present, the function tries to align data types between the data and the valueset (e.g., converting character codes to integers if necessary).

Value

A `tibble` with the same data as `data`, but with added attributes:

- Variable labels (via the `label` attribute)
- Value labels (as a `haven::labelled` class, if applicable)

Examples

```
data <- data.frame(
  sex = c(1, 2, 1),
  age = c(23, 45, 34)
)

metadata <- data.frame(
  variable_name = c("sex", "age"),
  label = c("Gender", "Age in years"),
  type = c("categorical", "numeric"),
  valueset = I(list(
    data.frame(value = c(1, 2), label = c("Male", "Female")),
    NULL
  ))
)

labelled_data <- add_metadata(data, metadata)
```

```
str(labelled_data)
```

---

as_rcdf *Convert to* rcdf *class*

Description

Converts an existing list or compatible object into an object of class rcdf.

Usage

```
as_rcdf(data)
```

Arguments

data	A list or object to be converted to class rcdf.

Value

The input object with class set to rcdf.

Examples

```
my_list <- list(a = 1, b = 2)
rcdf_obj <- as_rcdf(my_list)
class(rcdf_obj)
```

---

collect *Collect a lazy RCDF table into a data frame*

Description

Materialises a lazy rcdf_tbl_db DuckDB-backed table into a regular R data frame, optionally applying the variable labels and value labels stored in the table's metadata dictionary.

Usage

```
collect(data, ...)

## S3 method for class 'rcdf_tbl_db'
collect(data, ...)
```

Arguments

data	A lazy rcdf_tbl_db object (returned by read_rcdf with lazy = TRUE), or any object supported by dplyr::collect().
...	Additional arguments passed to dplyr::collect().

Value

A `tibble` with all rows materialised. If the table carries a metadata dictionary, variable labels and value labels are applied via `add_metadata` before returning.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path  <- file.path(dir, "mtcars.rcdf")
prv_key    <- file.path(dir, "sample-private-key-pw.pem")

## Not run:
result <- read_rcdf(path = rcdf_path, decryption_key = prv_key,
                    password = "1234", lazy = TRUE)
df <- collect(result$mtcars)
class(df)  # "tbl_df"

## End(Not run)
```

`decrypt_string` *Decrypt string using RSA*

Description

Decrypt string using RSA

Usage

```
decrypt_string(x, prv_key, password = NULL)
```

Arguments

`x`	Encrypted base64-encoded string
`prv_key`	A private key object or .pem file
`password`	Passwor of the private key

Value

A decrpyted character of length 1

Examples

```
dir <- system.file("extdata", package = "rcdf")
pub_key <- file.path(dir, 'sample-public-key.pem')
prv_key <- file.path(dir, 'sample-private-key.pem')
x <- encrypt_string('hello', pub_key)
decrypt_string(x, prv_key = prv_key, password = '1234')
```

encrypt_string *Encrypt string using RSA*

Description

Encrypt string using RSA

Usage

```
encrypt_string(x, pub_key)
```

Arguments

x	A character of length 1
pub_key	A public key object or .pem file

Value

Encrypted base64-encoded string

Examples

```
dir <- system.file("extdata", package = "rcdf")
pub_key <- file.path(dir, 'sample-public-key.pem')
encrypt_string('hello', pub_key)
```

generate_pw *Generate a random password*

Description

This function generates a random password of a specified length. It includes alphanumeric characters by default and can optionally include special characters.

Usage

```
generate_pw(length = 16, special_chr = TRUE)
```

Arguments

length	Integer. The length of the password to generate. Default is 16.
special_chr	Logical. Whether to include special characters (e.g., !, @, #, etc.) in the password. Default is TRUE.

Value

A character string representing the generated password.

Examples

```
generate_pw()
generate_pw(32)
generate_pw(12, special_chr = FALSE)
```

---

generate_rsa_keys *Generate RSA key pair and save to files*

Description

This function generates an RSA key pair (public and private) and saves them to specified files.

Usage

```
generate_rsa_keys(path, ..., password = NULL, which = "public", prefix = NULL)
```

Arguments

path	A character string specifying the directory path where the key files in .pem format should be saved.
...	Additional arguments passed to the openssl::rsa_keygen() function, such as key size.
password	A character string specifying the password for the private key. If NULL, the private key will not be encrypted.
which	A character string specifying which key to return. Can be either "public" or "private". Default is "public".
prefix	A character string used as a prefix for the key file names. Defaults to NULL, which will result in no prefix.

Value

A character string representing the file path of the generated key (either public or private, based on the which argument).

Examples

```
# Generate both public and private RSA keys and save them to the temp directory
path_to <- tempdir()
generate_rsa_keys(path = path_to, password = "securepassword")
```

get_attr *Get metadata attribute from RCDF data*

Description

Get metadata attribute from RCDF data

Usage

```
get_attr(rcdf, attr)
```

Arguments

rcdf	RCDF data
attr	Valid metadata key.

Value

RCDF attribute/s or NULL

Examples

```
## Not run:
# Assuming `df` is a valid RCDF object

get_attr(df, "area_names")

# To get nested attributes
get_attr(df, "meta.source_note")

## End(Not run)
```

get_attrs *Read all top-level metadata from an RCDF file*

Description

Extracts and returns the complete metadata JSON stored inside an `.rcdf` archive without decrypting or loading the underlying data. Useful for inspecting provenance, checksums, encryption parameters, and embedded data dictionaries without a decryption key.

Usage

```
get_attrs(path)
```

Arguments

path	Character string. Path to a valid .rcdf file.

Value

A named list corresponding to the metadata.json stored inside the archive. Common keys include log_id, created_at, version, checksum, dictionary, and key.

See Also

get_rcdf_metadata for retrieving a single metadata key, get_attr for reading metadata attributes attached to an in-memory RCDF object.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, "mtcars.rcdf")

meta <- get_attrs(rcdf_path)
meta$version
meta$created_at
```

---

get_rcdf_metadata *Extract metadata from an RCDF file*

Description

Retrieves a specific metadata value from a .rcdf file.

Usage

```
get_rcdf_metadata(path, name = NULL, key)
```

Arguments

path	Character string. The file path to the .rcdf file.
name	Character string. The metadata key to extract from the file.
key	**[Deprecated]** Character string. The metadata key to extract from the file.

Value

The value associated with the specified metadata key, or NULL if the key does not exist.

Examples

```
## Not run:
# Assuming "example.rcdf" is a valid RCDF file in the working directory:
get_rcdf_metadata("example.rcdf", "log_id")

## End(Not run)
```

`merge_rcdf` *Merge multiple RCDF files*

Description

Merge multiple RCDF files

Usage

```
merge_rcdf(
  rcdf_files,
  decryption_keys,
  passwords,
  merged_file_path,
  pub_key = NULL
)
```

Arguments

`rcdf_files`	A character vector of RCDF file paths
`decryption_keys`	Decryption keys associated with each RCDF file. Must match the length of the vector passed in the `rcdf_files` argument.
`passwords`	Password of the associated decryption keys. Must match the length of `decryption_keys`.
`merged_file_path`	File path or name of the merged RCDF file.
`pub_key`	Public key to encrypt the merged file. If `NULL`, a new RSA key pair will be generated.

Value

`NULL` (void)

Examples

```
## Not run:
dir <- system.file("extdata", package = "rcdf")

rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')
```

```
pw <- '1234'

temp_dir <- tempdir()

merge_rcdf(
  rcdf_files = rcdf_path,
  decryption_keys = private_key,
  passwords = pw,
  merged_file_path = file.path(temp_dir, "merged.rcdf"),
  pub_key = file.path(dir, 'sample-public-key-pw.pem')
)

unlink(file.path(temp_dir, "merged.rcdf"), force = TRUE)

## End(Not run)
```

`rcdf_list` *Create an empty* `rcdf` *object*

Description

Initializes and returns an empty `rcdf` object. This is a convenient constructor for creating a new `rcdf`-class list structure.

Usage

```
rcdf_list(...)
```

Arguments

`...`	Optional elements to include in the list. These will be passed to the internal list constructor and included in the resulting `rcdf` object.

Value

A list object of class `rcdf`.

Examples

```
rcdf <- rcdf_list()
class(rcdf)
```

read_dot_env *Read environment variables from a file*

Description

Based on https://github.com/gaborcsardi/dotenv

Usage

```
read_dot_env(path = ".env")
```

Arguments

path	A string specifying the path to the .env file. If not provided, defaults to .env in the current working directory.

Details

Reads a .env file containing environment variables in the format KEY=VALUE, and returns them as a named list. Lines starting with # are considered comments and ignored.

Value

A named list of environment variables. Each element is a key-value pair extracted from the file. If no variables are found, NULL is returned.

Examples

```
## Not run:
# Assuming an `.env` file with the following content:
# DB_HOST=localhost
# DB_USER=root
# DB_PASS="secret"

env_vars <- read_dot_env(".env")
print(env_vars)
# Should output something like:
# $DB_HOST
# [1] "localhost"

# If no path is given, it defaults to `.env` in the current directory.
env_vars <- read_dot_env()

## End(Not run)
```

read_env *Read environment variables from a file*

Description

[Deprecated]

Usage

```
read_env(path = ".env")
```

Arguments

path	A string specifying the path to the .env file. If not provided, defaults to .env in the current working directory.

Details

Reads a .env file containing environment variables in the format KEY=VALUE, and returns them as a named list. Lines starting with # are considered comments and ignored.

Value

A named list of environment variables. Each element is a key-value pair extracted from the file. If no variables are found, NULL is returned.

Examples

```
## Not run:
# Assuming an `.env` file with the following content:
# DB_HOST=localhost
# DB_USER=root
# DB_PASS="secret"

env_vars <- read_env(".env")
print(env_vars)
# Should output something like:
# $DB_HOST
# [1] "localhost"

# If no path is given, it defaults to `.env` in the current directory.
env_vars <- read_env()

## End(Not run)
```

read_parquet *Read Parquet file with optional decryption*

Description

This function reads a Parquet file, optionally decrypting it using the provided decryption key. If no decryption key is provided, it reads the file normally without decryption. It supports reading Parquet files as Arrow tables or regular data frames, depending on the as_arrow_table argument.

Usage

```
read_parquet(
  path,
  ...,
  decryption_key = NULL,
  as_arrow_table = FALSE,
  metadata = NULL
)
```

Arguments

path	The file path to the Parquet file.
...	Additional arguments passed to arrow::open_dataset() when no decryption key is provided.
decryption_key	A list containing aes_key and aes_iv. If provided, the Parquet file will be decrypted using these keys. Default is NULL.
as_arrow_table	Logical. If TRUE, the function will return the result as an Arrow table. If FALSE, a regular data frame will be returned. Default is FALSE.
metadata	Optional metadata (e.g., a data dictionary) to be applied to the resulting data.

Value

An Arrow table or a data frame, depending on the value of as_arrow_table.

Examples

```
## Not run:
# Using sample Parquet files from `mtcars` dataset
dir <- system.file("extdata", package = "rcdf")

# Not encrypted
read_parquet(file.path(dir, "mtcars.parquet"))

# Encrypted
read_parquet(
  file.path(dir, "mtcars-encrypted.parquet"),
  decryption_key = 'rppqM5CuEqotys4wQq/g7xh6wpIjRozcAIbI9sagwKE='
```

```
)

## End(Not run)
```

`read_parquet_tbl` *Read Parquet file as database*

Description

This function reads a Parquet file, optionally decrypting it using the provided decryption key. If no decryption key is provided, it reads the file normally without decryption. It supports reading Parquet files as Arrow tables or regular data frames, depending on the `as_arrow_table` argument.

Usage

```
read_parquet_tbl(conn, file, decryption_key, table_name = NULL, columns = NULL)
```

Arguments

`conn`	A DuckDB connection.
`file`	The file path to the Parquet file.
`decryption_key`	A list containing `aes_key` and `aes_iv`. If provided, the Parquet file will be decrypted using these keys. Default is `NULL`.
`table_name`	Database table name. If `NULL`, file name will be used as table name.
`columns`	A character vector matching the column names available in the Parquet file.

Value

Lazy table from DuckDB connection

Examples

```
## Not run:
# Using sample Parquet files from `mtcars` dataset
dir <- system.file("extdata", package = "rcdf")

# Encrypted
read_parquet_tbl(
  file.path(dir, "mtcars-encrypted.parquet"),
  decryption_key = 'rppqM5CuEqotys4wQq/g7xh6wpIjRozcAIbI9sagwKE='
)

## End(Not run)
```

read_rcdf *Read and decrypt RCDF data*

Description

This function reads an RCDF file, decrypts its contents using the specified decryption key, and loads it into R as an RCDF object.

Usage

```
read_rcdf(
  path,
  ...,
  decryption_key,
  password = NULL,
  metadata = list(),
  ignore_duplicates = TRUE,
  recursive = FALSE,
  return_meta = FALSE,
  lazy = FALSE,
  n_threads = NULL
)
```

Arguments

path	A string specifying the path to the RCDF archive (zip file). If a directory is provided, all `.rcdf` files within that directory will be processed.
...	Additional parameters passed to other functions, if needed (not yet implemented).
decryption_key	The key used to decrypt the RCDF. This can be an RSA or AES key, depending on how the RCDF was encrypted.
password	A password used for RSA decryption (optional).
metadata	An optional list of metadata object containing data dictionaries, value sets, and primary key constraints for data integrity measure (a `data.frame` or `tibble` that includes at least two columns: `file` and `pk_field_name`. This metadata is applied to the data if provided.
ignore_duplicates	A `logical` flag. If `TRUE`, a warning is issued when duplicates are found, based on the primary key/s defined during creation of RCDF file. If `FALSE`, the function stops with an error.
recursive	Logical. If `TRUE` and `path` is a directory, the function will search recursively for `.rcdf` files.
return_meta	Logical. If `TRUE`, the metadata will be returned as an attribute of the RCDF object.

lazy	Logical. If TRUE, each dataset is returned as a lazy rcdf_tbl_db DuckDB-backed table instead of being collected into memory. The underlying DuckDB connection is kept alive; call collect() on each element when you are ready to materialise. Default is FALSE.
n_threads	Integer or NULL. Number of DuckDB threads to use for reading. NULL (default) lets DuckDB choose based on available cores.

Value

An RCDF object, which is a list of Parquet files (one for each record) along with attached metadata.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')
pw <- '1234'

## Not run:
rcdf_data <- read_rcdf(
  path = rcdf_path,
  decryption_key = private_key,
  password = pw
)

rcdf_data

## End(Not run)
```

---

write_parquet *Write Parquet file with optional encryption*

Description

Writes a data frame to a Parquet file. When encryption_key is supplied the file is encrypted with AES using DuckDB's native Parquet encryption support. Without a key the file is written by arrow::write_parquet() and supports any compression codec understood by the Arrow library.

Usage

```
write_parquet(
  data,
  path,
  ...,
  encryption_key = NULL,
  conn = NULL,
  compression = "zstd"
)
```

Arguments

data	A data frame or tibble to write.
path	Character string. Destination file path for the Parquet file.
...	Additional arguments passed to arrow::write_parquet() when encryption_key is NULL.
encryption_key	A raw AES key as a hex string (32, 48, or 64 hex characters) or a base-64–encoded 256-bit key string. When NULL (default) no encryption is applied.
conn	An optional existing DuckDB DBIConnection. When supplied, the connection is reused instead of opening a new one, which avoids per-call setup overhead when writing many tables in a batch. The caller is responsible for disconnecting the connection. Default is NULL.
compression	Compression codec to use for unencrypted Parquet files. Any codec supported by arrow::write_parquet() is valid (e.g. "zstd", "snappy", "gzip", "lz4", "uncompressed"). Ignored when encryption_key is set (DuckDB chooses the codec). Default is "zstd".

Value

NULL invisibly. The Parquet file is written to path.

Examples

```
## Not run:
library(rcdf)

data <- mtcars
key <- "rppqM5CuEqotys4wQq/g7xh6wpIjRozcAIbI9sagwKE="

temp_dir <- tempdir()

# Encrypted write
write_parquet(
  data = data,
  path = file.path(temp_dir, "mtcars.parquet"),
  encryption_key = key
)

# Unencrypted write with gzip compression
write_parquet(
  data = data,
  path = file.path(temp_dir, "mtcars-gz.parquet"),
  compression = "gzip"
)

## End(Not run)
```

`write_rcdf` *Write data to RCDF format*

Description

This function writes data to an RCDF (Reusable Data Container Format) archive. It encrypts the data using AES, generates metadata, and then creates a zip archive containing both the encrypted Parquet files and metadata. The function supports the inclusion of metadata such as system information and encryption keys.

Usage

```
write_rcdf(
  data,
  path,
  pub_key,
  ...,
  metadata = list(),
  ignore_duplicates = TRUE
)
```

Arguments

`data`	A list of data frames or tables to be written to RCDF format. Each element of the list represents a record.
`path`	The path where the RCDF file will be written. The file will be saved with a `.rcdf` extension if not already specified.
`pub_key`	The public RSA key used to encrypt the AES encryption keys.
`...`	Additional arguments passed to helper functions if needed.
`metadata`	A list of metadata to be included in the RCDF file.
`ignore_duplicates`	A `logical` flag. If `TRUE`, a warning is issued when duplicates are found. If `FALSE`, the function stops with an error.

Value

NULL. The function writes the data to a `.rcdf` file at the specified path.

Examples

```
## Not run:
# Example usage of writing an RCDF file

rcdf_data <- rcdf_list()
rcdf_data$mtcars <- mtcars

dir <- system.file("extdata", package = "rcdf")
```

```
temp_dir <- tempdir()

write_rcdf(
  data = rcdf_data,
  path = file.path(temp_dir, "mtcars.rcdf"),
  pub_key = file.path(dir, 'sample-public-key.pem')
)

write_rcdf(
  data = rcdf_data,
  path = file.path(temp_dir, "mtcars-pw.rcdf"),
  pub_key = file.path(dir, 'sample-public-key-pw.pem')
)

unlink(file.path(temp_dir, "mtcars.rcdf"), force = TRUE)
unlink(file.path(temp_dir, "mtcars-pw.rcdf"), force = TRUE)

## End(Not run)
```

`write_rcdf_as` *Write RCDF data to multiple formats*

Description

Exports RCDF-formatted data to one or more supported open data formats. The function automatically dispatches to the appropriate writer function based on the `formats` provided.

Usage

```
write_rcdf_as(data, path, formats, ...)
```

Arguments

`data`	A named list or RCDF object. Each element should be a table or tibble-like object (typically a `dbplyr` or `dplyr` table).
`path`	The target directory where output files should be saved.
`formats`	A character vector of file formats to export to. Supported formats include: `"csv"`, `"tsv"`, `"json"`, `"parquet"`, `"xlsx"`, `"dta"`, `"sav"`, and `"sqlite"`.
`...`	Additional arguments passed to the respective writer functions.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_csv write_rcdf_tsv write_rcdf_json write_rcdf_xlsx write_rcdf_dta write_rcdf_sav write_rcdf_sqlite

Examples

```
## Not run:
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_as(data = rcdf_data, path = temp_dir, formats = c("csv", "xlsx"))

unlink(temp_dir, force = TRUE)

## End(Not run)
```

`write_rcdf_csv` *Write RCDF data to CSV files*

Description

Writes each table in the RCDF object as a separate `.csv` file.

Usage

```
write_rcdf_csv(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The base output directory.
`...`	Additional arguments passed to `write.csv()`.
`parent_dir`	Optional subdirectory under `path` to group CSV files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_csv(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_dta` *Write RCDF data to Stata* `.dta` *files*

Description

Writes each table in the RCDF object to a `.dta` file for use in Stata.

Usage

```
write_rcdf_dta(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for files.
`...`	Additional arguments passed to `haven::write.dta()`.
`parent_dir`	Optional subdirectory under `path` to group Stata files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_dta(data = rcdf_data, path = temp_dir)
```

```
unlink(temp_dir, force = TRUE)
```

`write_rcdf_json` *Write RCDF data to JSON files*

Description

Writes each table in the RCDF object as a separate `.json` file.

Usage

```
write_rcdf_json(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The output directory for files.
`...`	Additional arguments passed to `jsonlite::write_json()`.
`parent_dir`	Optional subdirectory under `path` to group JSON files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_json(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_parquet` *Write RCDF data to Parquet files*

Description

This function writes an RCDF object (a list of data frames) to multiple Parquet files. Each data frame in the list is written to its corresponding Parquet file in the specified path.

Usage

```
write_rcdf_parquet(
  data,
  path,
  ...,
  parent_dir = NULL,
  primary_key = NULL,
  ignore_duplicates = TRUE
)
```

Arguments

`data`	A list where each element is a data frame or tibble that will be written to a Parquet file.
`path`	The directory path where the Parquet files will be written.
`...`	Additional arguments passed to `rcdf::write_parquet()` while writing each Parquet file.
`parent_dir`	An optional parent directory to be included in the path where the files will be written.
`primary_key`	A `data.frame` or `tibble` that includes at least two columns: `file` and `pk_field_name`.
`ignore_duplicates`	A `logical` flag. If `TRUE`, a warning is issued when duplicates are found. If `FALSE`, the function stops with an error.

Value

A character vector of file paths to the written Parquet files.

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_parquet(data = rcdf_data, path = temp_dir)
```

```
unlink(temp_dir, force = TRUE)
```

`write_rcdf_sav` *Write RCDF data to SPSS* `.sav` *files*

Description

Writes each table in the RCDF object to a `.sav` file using the haven package for compatibility with SPSS.

Usage

```
write_rcdf_sav(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for files.
`...`	Additional arguments passed to `haven::write_sav()`.
`parent_dir`	Optional subdirectory under `path` to group SPSS files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_sav(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_sqlite` *Write RCDF data to a SQLite database*

Description

Writes all tables in the RCDF object to a single SQLite database file.

Usage

```
write_rcdf_sqlite(data, path, db_name = "cbms_data", ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	Output directory for the database file.
`db_name`	Name of the SQLite database file (without extension).
`...`	Additional arguments passed to `DBI::dbWriteTable()`.
`parent_dir`	Optional subdirectory under `path` to store the SQLite file.

Value

Invisibly returns `NULL`. A `.db` file is written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_sqlite(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_tsv` *Write RCDF data to TSV files*

Description

Writes each table in the RCDF object as a separate tab-separated `.txt` file.

Usage

```
write_rcdf_tsv(data, path, ..., parent_dir = NULL)
```

Arguments

`data`	A valid RCDF object.
`path`	The base output directory.
`...`	Additional arguments passed to `write.table()`.
`parent_dir`	Optional subdirectory under `path` to group TSV files.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_tsv(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

`write_rcdf_xlsx` *Write RCDF data to Excel files*

Description

Writes each table in the RCDF object as a separate `.xlsx` file using the `openxlsx` package.

Usage

```
write_rcdf_xlsx(
  data,
  path,
  ...,
  parent_dir = NULL,
  as_single_file = FALSE,
  file_name = NULL
)
```

Arguments

`data`	A valid RCDF object.
`path`	The output directory.
`...`	Additional arguments passed to `openxlsx::write.xlsx()`.
`parent_dir`	Optional subdirectory under `path` to group Excel files.
`as_single_file`	Whether to export all records (items in the RCDF list) in a single file where each item will be written per sheet in the workbook.
`file_name`	File name to assign when `as_single_file` is set to `TRUE`.

Value

Invisibly returns `NULL`. Files are written to disk.

See Also

write_rcdf_as

Examples

```
dir <- system.file("extdata", package = "rcdf")
rcdf_path <- file.path(dir, 'mtcars.rcdf')
private_key <- file.path(dir, 'sample-private-key-pw.pem')

rcdf_data <- read_rcdf(path = rcdf_path, decryption_key = private_key, password = '1234')
temp_dir <- tempdir()

write_rcdf_xlsx(data = rcdf_data, path = temp_dir)

unlink(temp_dir, force = TRUE)
```

Index