

Package: R6.tiledb (via r-universe)

October 13, 2026

Title R6 Classes for TileDB Arrays and Groups

Version 0.3.4

Date 2026-10-12

Description Provides foundational R6 classes to interface with 'TileDB' storage engine using 'tiledb' package. The goal is to offer a set of minimum classes based on the R6 object oriented system in order to make it easier to work with 'TileDB' Arrays and Groups. The base classes 'TileDBArray' and 'TileDBGroup' can be easily extended to new subclasses to represent and support any data model.

License MIT + file LICENSE

URL <https://github.com/cgiachalis/R6.tiledb>,
<https://cgiachalis.github.io/R6.tiledb/>

BugReports <https://github.com/cgiachalis/R6.tiledb/issues>

Depends R (>= 4.4.0)

Imports bit64, cli, fs, methods, R6, spd, tiledb (>= 0.33.0)

Suggests litedown, mirai (>= 2.4.0), data.table, nanotime, spelling, testthat (>= 3.0.0), withr

VignetteBuilder litedown

Remotes TileDB-Inc/TileDB-R@v0.33.1

Config/testthat/edition 3

Config/testthat/start-first TileDBGroup

Encoding UTF-8

Language en-GB

Roxygen list(markdown = TRUE, old_usage = TRUE)

Config/roxygen2/version 8.1.0

Config/pak/sysreqs cmake git make libuv1-dev libzstd-dev

Repository <https://cgiachalis.r-universe.dev>

Date/Publication 2026-10-13 09:20:58 UTC

RemoteUrl <https://github.com/cgiachalis/R6.tiledb>
RemoteRef HEAD
RemoteSha 80a3bd54f64257496bf80d7c47bd88af5a3c61ee

Contents

array_timestamps	2
close	3
delete_metadata	4
demo_array_UCBAdmissions	5
fetch_metadata	5
group_timestamps	6
metadata	7
open_write	9
set_config_params	10
set_consolidation_timestamps	11
set_group_timestamps	12
set_metadata	13
set_tiledb_timestamp	14
tdb_array	15
tdb_array_create	16
tdb_fragments	17
tdb_group	18
tdb_group_create	20
TileDBArray	20
TileDBArrayExp	23
TileDBFragments	30
TileDBGroup	33
TileDBGroupExp	37
vfs_dir_tree	39
vfs_size	40

Index **41**

array_timestamps	<i>Get Array Timestamps</i>
------------------	-----------------------------

Description

Get Array Timestamps

Usage

```
## S3 method for class 'tiledb_array'
array_timestamps(object, tz = "", ...)

## S3 method for class 'TileDBArray'
array_timestamps(object, tz = "", ...)
```

Arguments

<code>object</code>	An R object that contains a TileDB resource pointer.
<code>tz</code>	A character string for the time zone specification to be used for the conversion in print method only. Defaults to Sys.timezone() .
<code>...</code>	Other arguments passed to methods. Not used.

Value

An object of class `array_timestamps` that is a list that holds the user supplied temporal timestamps and the time range the array is opened at.

See Also

[group_timestamps\(\)](#), [set_tiledb_timestamp\(\)](#)

close	<i>Close a TileDB Resource</i>
-------	--------------------------------

Description

Close a [tiledb::tiledb_array\(\)](#), [tiledb::tiledb_group\(\)](#), [TileDBArray](#) or [TileDBGroup](#) object.

Usage

```
## S3 method for class 'tiledb_array'
close(con, ...)

## S3 method for class 'TileDBArray'
close(con, ...)

## S3 method for class 'tiledb_group'
close(con, ...)

## S3 method for class 'TileDBGroup'
close(con, ...)
```

Arguments

<code>con</code>	An R object that contains a TileDB resource pointer.
<code>...</code>	Other arguments passed to methods. Not used.

Value

Invisibly, a logical TRUE on success.

delete_metadata	<i>Delete TileDB Metadata</i>
-----------------	-------------------------------

Description

Delete metadata using keys names for a TileDB array or group.

Usage

```
## S3 method for class 'TileDBArray'
delete_metadata(x, keys, ...)

## S3 method for class 'TileDBGroup'
delete_metadata(x, keys, ...)

## S3 method for class 'tiledb_array'
delete_metadata(x, keys, ...)

## S3 method for class 'tiledb_group'
delete_metadata(x, keys, ...)

## S3 method for class 'character'
delete_metadata(x, keys, ctx = NULL, ...)
```

Arguments

x	An R object that points to a TileDB resource whose metadata are to be accessed.
keys	A character vector of metadata key names to be accessed.
...	Other arguments passed to methods. Not used.
ctx	Optional <code>tiledb::tiledb_ctx()</code> object.

Details

`delete_metadata()` works similar to `metadata<-()` with NULL value but works with a character vector of keys.

The character method requires a valid URI path.

The methods will not alter the mode of the TileDB object; also, the object will be opened temporarily to access the metadata if it is closed.

Value

A logical TRUE, invisibly.

See Also

`set_metadata()` and `metadata()`.

demo_array_UCBAdmissions

Create a demo array for UCBAdmissions dataset

Description

Create a demo array for UCBAdmissions dataset

Usage

```
demo_array_UCBAdmissions(uri)
```

Arguments

uri The uri path.

fetch_metadata

Fetch TileDB Metadata

Description

Fetch metadata using a list of key-value pairs for a TileDB array or group. Optionally, you can access metadata at specific point in time (time-travelling).

Usage

```
## S3 method for class 'TileDBArray'
fetch_metadata(x, keys = NULL, timestamp = NULL, ...)
```

```
## S3 method for class 'TileDBGroup'
fetch_metadata(x, keys = NULL, timestamp = NULL, ...)
```

```
## S3 method for class 'tiledb_array'
fetch_metadata(x, keys = NULL, timestamp = NULL,
  ...)
```

```
## S3 method for class 'tiledb_group'
fetch_metadata(x, keys = NULL, timestamp = NULL,
  ...)
```

```
## S3 method for class 'character'
fetch_metadata(x, keys = NULL, timestamp = NULL,
  ctx = NULL, ...)
```

Arguments

x	An R object that points to a TileDB resource whose metadata are to be accessed.
keys	A character vector of metadata key names to be accessed. When NULL (default) all metadata will be accessed.
timestamp	Optional datetime object of class "POSIXct" to write at this timestamp.
...	Other arguments passed to methods. Not used.
ctx	Optional <code>tiledb::tiledb_ctx()</code> object.

Details

`fetch_metadata()` works similar to `metadata()` but works with a list of key value pairs and returns a list always.

The character method requires a valid URI path.

The methods will not alter the mode of the TileDB object; also, the object will be opened temporarily to access the metadata if it is closed.

Value

A named list of class `tdb_metadata`.

See Also

`set_metadata()` and `metadata()`.

group_timestamps	<i>Get Group Timestamps</i>
------------------	-----------------------------

Description

Get Group Timestamps

Usage

```
## S3 method for class 'tiledb_ctx'
group_timestamps(object, tz = "", ...)

## S3 method for class 'tiledb_group'
group_timestamps(object, tz = "", ...)

## S3 method for class 'tiledb_config'
group_timestamps(object, tz = "", ...)

## S3 method for class 'TileDBGroup'
group_timestamps(object, tz = "", from = c("ctx",
  "cfg"), ...)
```

Arguments

object	An R object that contains a TileDB resource pointer.
tz	A character string for the time zone specification to be used for the conversion in print method only. Defaults to Sys.timezone() .
...	Other arguments passed to methods. Not used.
from	TileDBGroup's source of timestamps: either from context ctx, or group object configuration cfg.

Value

An object of class `group_timestamps` that is a list with the time range the group is opened at.

See Also

[array_timestamps\(\)](#), [set_tiledb_timestamp\(\)](#)

metadata	<i>Get, Set a TileDB Metadata Key</i>
----------	---------------------------------------

Description

Get or set a metadata key for a TileDB array or group.

Usage

```
## S3 method for class 'TileDBArray'
metadata(x, which, ...)

## S3 method for class 'TileDBGroup'
metadata(x, which, ...)

## S3 method for class 'tiledb_array'
metadata(x, which, ...)

## S3 method for class 'tiledb_group'
metadata(x, which, ...)

## S3 method for class 'character'
metadata(x, which, ctx = NULL, ...)

## S3 replacement method for class 'TileDBArray'
metadata(x, which, ...) <- value

## S3 replacement method for class 'TileDBGroup'
metadata(x, which, ...) <- value
```

```
## S3 replacement method for class 'tiledb_array'
metadata(x, which, ...) <- value

## S3 replacement method for class 'tiledb_group'
metadata(x, which, ...) <- value

## S3 replacement method for class 'character'
metadata(x, which, ctx = NULL, ...) <- value
```

Arguments

<code>x</code>	An R object that points to a TileDB resource whose metadata are to be accessed.
<code>which</code>	A non-empty character string specifying which metadata key is to be accessed.
<code>...</code>	Other arguments passed to methods. Not used.
<code>ctx</code>	Optional <code>tiledb::tiledb_ctx()</code> object.
<code>value</code>	An object, the new value of the metadata, or <code>NULL</code> to remove the key. Note that character vectors should be of length one (scalar).

Details

These functions work similar to `attr()` and provide access to a single metadata key on an TileDB object. The replacement form (setter) puts a metadata key with the value specified or creates a new metadata with the value given.

The character method is intended for a valid URI path.

The methods will not alter the mode of the TileDB object.

For the extractor the object will be opened temporarily to access the metadata if it is closed.

For the replacement, the object will be opened in write mode temporarily and on exit will revert back to previous mode; in the case where the object is already in write mode then it will be reopened at the same mode in order to flush the metadata on disk.

Value

For the extractor, the key value of the metadata matched, or `NULL` if no exact match is found.

See Also

For a list of metadata and time-travelling use `set_metadata()` and `fetch_metadata()`.

open_write

*Open a TileDB Resource at Write Mode***Description**

Open a URI, `tiledb::tiledb_array()`, `tiledb::tiledb_group()`, `TileDBArray` or `TileDBGroup` at write mode with an optional timestamp.

Usage

```
## S3 method for class 'TileDBArray'
open_write(object, timestamp = NULL, ctx = NULL, ...)

## S3 method for class 'tiledb_array'
open_write(object, timestamp = NULL, ctx = NULL,
  ...)

## S3 method for class 'TileDBGroup'
open_write(object, timestamp = NULL, ctx = NULL, ...)

## S3 method for class 'tiledb_group'
open_write(object, timestamp = NULL, ctx = NULL,
  ...)

## S3 method for class 'character'
open_write(object, timestamp = NULL, ctx = NULL, ...)
```

Arguments

<code>object</code>	An R object that contains a TileDB resource pointer.
<code>timestamp</code>	Optional datetime object of class "POSIXct" to write at this timestamp.
<code>ctx</code>	Optional <code>tiledb::tiledb_ctx()</code> object.
<code>...</code>	Other arguments passed to methods. Not used.

Details

Opening a character string should be a valid URI path for a TileDB resource.

Objects other than a URI character are implicitly closed if found opened and re-opened at write mode.

Note that when using the `timestamp` argument, the `TileDBArray`, `TileDBGroup` and their sub-classes will have different timestamps from the returning TileDB object. This is by design, you can not write at timestamp using `TileDBArray` and `TileDBGroup` interface only via `open_write()` method.

Value

An object of class `tiledb_array` or `tiledb_group` depending on the method; the object is opened in 'WRITE' mode.

set_config_params	<i>Modify Config Parameters</i>
-------------------	---------------------------------

Description

These functions will set or unset the parameters of TileDB Config object and return the modified configuration object.

Usage

```
set_config_params(cfg, keyval)
```

```
unset_config_params(cfg, keys)
```

Arguments

<code>cfg</code>	A configuration object <code>tiledb::tiledb_config()</code> .
<code>keyval</code>	A named character vector with configuration parameters.
<code>keys</code>	A character with configuration keys to unset.

Value

The modified `tiledb_config` object.

See Also

`tiledb::tiledb_config()`

Examples

```
origparams <- c("sm.consolidation.timestamp_start" = "100",
               "sm.consolidation.timestamp_end" = "20000")

# Create a context with a config setting the params above
cfg <- tiledb::tiledb_config(origparams)
ctx <- tiledb::tiledb_ctx(config = cfg)

# Do something with ctx

# Get configs from context
cfg <- tiledb::config(ctx)

# New params
params <- c("sm.consolidation.timestamp_start" = "100",
```

```

    "sm.consolidation.timestamp_end" = "10000")

cfg <- set_config_params(cfg, params)

cfg["sm.consolidation.timestamp_end"] # "10000"

# Update context with modified config object
ctx <- tiledb::tiledb_ctx(config = cfg)

# Reset consolidation timestamps
cfg <- unset_config_params(cfg, names(params))

# check the default values
cfg["sm.consolidation.timestamp_start"] # "0"
cfg["sm.consolidation.timestamp_end"]   # "18446744073709551615"

# Update context with modified config object
ctx <- tiledb::tiledb_ctx(config = cfg)

```

set_consolidation_timestamps

Modify Consolidation Start/End Timestamps

Description

Set or unset consolidation timestamps to the given configuration object.

Usage

```

set_consolidation_timestamps(cfg, start_time = NULL, end_time = NULL)

unset_consolidation_timestamps(cfg)

```

Arguments

`cfg` A configuration object `tiledb::tiledb_config()`.
`start_time, end_time` Optional time stamp values. A date time objects of class POSIXt.

Value

The modified `tiledb_config` object.

See Also

[set_config_params\(\)](#)

Examples

```

cfg <- tiledb::tiledb_config()
cfg <- set_consolidation_timestamps(cfg, as.POSIXct(1), as.POSIXct(1000))
cfg["sm.consolidation.timestamp_start"] # 1000 (ms)

# reset
cfg <- unset_consolidation_timestamps(cfg)
cfg["sm.consolidation.timestamp_start"] # 0

```

set_group_timestamps *Modify Group Start/End Open Timestamps*

Description

Set or unset group opening start, end timestamps to the given configuration object.

Usage

```

set_group_timestamps(cfg, start_time = NULL, end_time = NULL)

unset_group_timestamps(cfg)

```

Arguments

cfg A configuration object `tiledb::tiledb_config()`.
start_time, end_time Optional time stamp values. A date time objects of class POSIXt.

Value

The modified tiledb_config object.

See Also

[set_config_params\(\)](#)

Examples

```

cfg <- tiledb::tiledb_config()
cfg <- set_group_timestamps(cfg, as.POSIXct(1), as.POSIXct(1000))
cfg["sm.group.timestamp_start"] # 1000 (ms)

# reset
cfg <- unset_group_timestamps(cfg)
cfg["sm.group.timestamp_start"] # 0

```

set_metadata

*Set TileDB Metadata***Description**

Set metadata using a list of key-value pairs for a TileDB array or group. Optionally, you can write metadata at specific point in time (time-travelling).

Usage

```
## S3 method for class 'TileDBArray'
set_metadata(x, keys, timestamp = NULL, ...)

## S3 method for class 'TileDBGroup'
set_metadata(x, keys, timestamp = NULL, ...)

## S3 method for class 'tiledb_array'
set_metadata(x, keys, timestamp = NULL, ...)

## S3 method for class 'tiledb_group'
set_metadata(x, keys, timestamp = NULL, ...)

## S3 method for class 'character'
set_metadata(x, keys, timestamp = NULL, ctx = NULL,
  ...)
```

Arguments

x	An R object that points to a TileDB resource whose metadata are to be written.
keys	A named list of key value metadata.
timestamp	Optional datetime object of class "POSIXct" to write at this timestamp.
...	Other arguments passed to methods. Not used.
ctx	Optional <code>tiledb::tiledb_ctx()</code> object.

Details

`set_metadata()` works similar to `metadata<-( )` but works with a list of key value pairs.

The optional argument `timestamp` can be used to set metadata at specific point in time.

The character method requires a valid URI path.

The methods will not alter the mode of the TileDB object.

The object will be opened in write mode temporarily and on exit will revert back to previous mode; in the case where the object is already in write mode then it will be reopened at the same mode in order to flush the metadata on disk.

Value

A logical TRUE, invisibly.

See Also

[fetch_metadata\(\)](#) and [metadata\(\)](#).

set_tiledb_timestamp *Set a TileDB timestamp*

Description

Define a list of start,end timestamps intended for opening a TileDB object, either at initialisation or via active field tiledb_timestamp.

Usage

```
set_tiledb_timestamp(start_time, end_time, tz = "")
```

Arguments

start_time, end_time

An object coercible to POSIXct. See [as.POSIXct\(\)](#).

tz

A character string for the time zone specification to be used for the conversion. Defaults to [Sys.timezone\(\)](#).

Value

An list object of class tiledb_timestamp.

See Also

[TileDBObject](#)

Examples

```
# Character input
set_tiledb_timestamp("2025-01-01", "2025-08-01", tz = "UTC")

# Numeric input
set_tiledb_timestamp(1000000, 1000000*4, tz = "UTC")

# Date end point
set_tiledb_timestamp(end_time = as.Date("1990-01-01"), tz = "UTC")

# Default
set_tiledb_timestamp(tz = "Europe/London")
```

```
# Invalid: start_time > end_time
# set_tiledb_timestamp(start_time = 1, end_time = 0)
# Error: `start_time` is greater than `end_time`.
```

tdb_array	<i>Open TileDBArrayExp</i>
-----------	----------------------------

Description

Functional interface that initialises a [TileDBArrayExp](#) instance and opens an array object at READ or WRITE mode.

Usage

```
tdb_array(uri, mode = "READ", ctx = NULL, tiledb_timestamp = NULL)
```

Arguments

uri	URI path for the TileDB object.
mode	Mode to open : either "READ" (default) or "WRITE".
ctx	Optional tiledb::tiledb_ctx() object.
tiledb_timestamp	Set a TileDB timestamp range that the resource will be opened at. Effective only for "READ" mode. Valid options: • A NULL value (default) • An R object coercible to POSIXct with length 1 which used for end timestamp, or length 2 with start, end timestamps • An object of class tiledb_timestamp. See set_tiledb_timestamp() Also, it can be set through active field \$tiledb_timestamp.

Value

A TileDBArrayExp, R6 object.

Active bindings

- ctx : A TileDB Context. See [tiledb::tiledb_ctx\(\)](#)
- tiledb_timestamp : A TileDB timestamp range that the resource will be opened at. See [set_tiledb_timestamp\(\)](#)
- uri : The URI of the TileDB object
- mode: Get the mode of the object: one of the following: "CLOSED", "READ" or "WRITE"
- object_type : The TileDB object type: "ARRAY", "GROUP" or "INVALID"
- object : Access the underlying [tiledb::tiledb_array\(\)](#) object. When used before open() method, the underlying array will be initialised at "READ" mode and kept open

- `fragments_object` : Access the [TileDBFragments](#) instance for this array
- `schema_version` : Retrieve the schema version for this array
- `is_sparse` : Check array schema for sparsity
- `size` : Directory size

Methods

Public Methods

Examples

```
## Not run:
uri <- tempdir()

# create demo array on disk
demo_array_UCBAdmissions(uri)

arrobj <- tdb_array(uri)

arrobj$object_type
#> "ARRAY"

arrobj$frag_num()
#> 3

arrobj$colnames()
#> "Dept" "Gender" "Admit" "Freq"

arrobj$has_enumeration()
#> Admit Freq
#> TRUE FALSE

## End(Not run)
```

tdb_array_create

Create a TileDB Array

Description

Given a TileDB schema, it will create an array and instantiate a `TileDBArrayExp` object. The array will be opened at the given mode and kept opened. It can be accessed via active field `$object`.

Usage

```
tdb_array_create(uri, sch, mode = "WRITE", ctx = NULL)
```

Arguments

uri	URI path for the TileDB object.
sch	A TileDB schema. See constructor tiledb::tiledb_array_schema() .
mode	Mode to open: either "READ" or "WRITE" (default).
ctx	A TileDB Context. See tiledb::tiledb_ctx() .

Value

A TileDBArrayExp, R6 object.

Examples

```
suppressMessages(library(tiledb))

# construct schema
schema <- tiledb_array_schema(
  dom = tiledb_domain(dims = tiledb_dim("id", c(1L, 4L), 4L, "INT32")),
  attrs = tiledb_attr("model", type = "ASCII"))

# uri path
uri <- tempfile()

# create and open array at WRITE mode
arrobj <- tdb_array_create(uri, sch = schema, mode = "WRITE")

arrobj$schema_info()

arrobj$mode # WRITE
```

tdb_fragments	<i>Open TileDBFragments</i>
---------------	-----------------------------

Description

Functional interface that initialises a [TileDBFragments](#) instance.

Usage

```
tdb_fragments(uri, ctx = NULL)
```

Arguments

uri	URI path for the TileDB object.
ctx	Optional tiledb::tiledb_ctx() object.

Value

A TileDBFragments, R6 object.

Active bindings

- uri : The URI of the TileDB object
- fragment_info : Get the TileDB Fragment Info object as returned by [tiledb::tiledb_fragment_info](#).

Methods

Public Methods

Examples

```
## Not run:
uri <- tempdir()
fobj <- demo_array_UCBAdmissions(uri)

# Get metadata information for first fragment
fobj$get_ifragment(1)

## End(Not run)
```

tdb_group	<i>Open TileDBGroupExp</i>
-----------	----------------------------

Description

Functional interface that initialises a [TileDBGroupExp](#) instance and opens a group object at READ or WRITE mode.

Usage

```
tdb_group(uri, mode = "READ", ctx = NULL, tiledb_timestamp = NULL)
```

Arguments

- | | |
|------------------|---|
| uri | URI path for the TileDB object. |
| mode | Mode to open : either "READ" (default) or "WRITE". |
| ctx | Optional tiledb::tiledb_ctx() object. |
| tiledb_timestamp | Set a TileDB timestamp range that the resource will be opened at. Effective only for "READ" mode. Valid options: <ul style="list-style-type: none">• A NULL value (default)• An R object coercible to POSIXct with length 1 which used for end timestamp, or length 2 with start, end timestamps |

- An object of class tiledb_timestamp. See [set_tiledb_timestamp\(\)](#)
- Also, it can be set through active field \$tiledb_timestamp.

Value

A TileDBGroupExp, R6 object.

Active bindings

- ctx : A TileDB Context. See [tiledb::tiledb_ctx\(\)](#)
- tiledb_timestamp : A TileDB timestamp range that the resource will be opened at. See [set_tiledb_timestamp\(\)](#)
- uri : The URI of the TileDB object
- mode: Get the mode of the object: one of the following: "CLOSED", "READ" or "WRITE"
- object_type : The TileDB object type: "ARRAY", "GROUP" or "INVALID"
- object : Access the underlying [tiledb::tiledb_group\(\)](#) object
- members: Access the list of group members.
- size : Directory size

Methods

Public Methods

Examples

```
## Not run:
# uri path
uri <- tempdir()

obj <- TileDBGroup$new(uri)

obj$create()

obj$close()

# new instance
newobj <- tdb_group(uri)

newobj$is_open() # TRUE

newobj$mode # "READ"

## End(Not run)
```

tdb_group_create	<i>Create a TileDB Group</i>
------------------	------------------------------

Description

Create a group and instantiate a TileDBGroupExp object. The group will be opened at the given mode and kept opened. It can be accessed via active field \$object.

Usage

```
tdb_group_create(uri, mode = "WRITE", ctx = NULL)
```

Arguments

uri	URI path for the TileDB object.
mode	Mode to open: either "READ" or "WRITE" (default).
ctx	A TileDB Context. See tiledb::tiledb_ctx() .

Value

A TileDBGroupExp, R6 object.

Examples

```
# uri path
uri <- tempfile()

# create and open array at WRITE mode
grpobj <- tdb_group_create(uri,mode = "WRITE")

grpobj$mode # WRITE

grpobj$count_members() # 0
```

TileDBArray	<i>Generate a TileDBArray Object</i>
-------------	--------------------------------------

Description

Base class for representing a TileDB Array.

Initialization:

A new TileDBArray instance is initialized using the new() method:

```
# uri path
uri <- tempdir()
# new instance
obj <- TileDBArray$new(uri = uri)
# does array exist at this uri
obj$exists() # FALSE

unlink(uri)
```

Value

An object of class `TileDBArray`, R6.

Super class

`TileDBObject` -> `TileDBArray`

Active bindings

`object` Access the underlying `tiledb::tiledb_array()` object. When used before `open()` method, the underlying array will be initialised at "READ" mode and kept open.

Methods**Public methods:**

- `TileDBArray$open()`
- `TileDBArray$close()`
- `TileDBArray$tiledb_array()`
- `TileDBArray$schema()`
- `TileDBArray$schema_info()`
- `TileDBArray$dimensions()`
- `TileDBArray$attributes()`
- `TileDBArray$dimnames()`
- `TileDBArray$attrnames()`
- `TileDBArray$colnames()`
- `TileDBArray$print()`
- `TileDBArray$clone()`

`TileDBArray$open()`: Open TileDB array object for read or write.

This methods opens the underlying `tiledb::tiledb_array()` object in the new mode if it is different from the current mode.

When the new mode and current mode is the same, no action is taken. To force close and then open again use `reopen()` method.

When a time-stamp is specified, it will be effective in "READ" mode only.

Usage:

```
TileDBArray$open(mode = c("READ", "WRITE"))
```

Arguments:

mode Mode to open : either "READ" or "WRITE". Default is "READ".

Returns: The object, invisibly.

TileDBArray\$close(): Close the object.

Usage:

```
TileDBArray$close()
```

Returns: The object, invisibly.

TileDBArray\$tiledb_array(): Return a [tiledb::tiledb_array](#) object.

If a query_type not provided then it will be inherited from class mode; in case the class mode is "CLOSED", then the query type defaults to "READ".

Usage:

```
TileDBArray$tiledb_array(...)
```

Arguments:

... Optional arguments to pass to [tiledb::tiledb_array\(\)](#).

Returns: A [tiledb::tiledb_array](#) object.

TileDBArray\$schema(): Retrieve the array schema as TileDB schema.

Usage:

```
TileDBArray$schema()
```

Returns: A [tiledb::tiledb_array_schema](#) object.

TileDBArray\$schema_info(): Retrieve schema information.

Usage:

```
TileDBArray$schema_info()
```

Returns: A data.frame object with four column schema information: names, types, status and enum.

TileDBArray\$dimensions(): Retrieve the array dimensions.

Usage:

```
TileDBArray$dimensions()
```

Returns: A named list of [tiledb::tiledb_dim](#) objects.

TileDBArray\$attributes(): Retrieve the array attributes.

Usage:

```
TileDBArray$attributes()
```

Returns: A list of [tiledb::tiledb_attr](#) objects.

TileDBArray\$dimnames(): Retrieve dimension names.

Usage:

```
TileDBArray$dimnames()
```

Returns: A character vector with the array's dimension names.

TileDBArray\$attrnames(): Retrieve attribute names.

Usage:

```
TileDBArray$attrnames()
```

Returns: A character vector with the array's attribute names.

TileDBArray\$colnames(): Retrieve the names of all columns, including dimensions and attributes.

Usage:

```
TileDBArray$colnames()
```

Returns: A character vector with the array's column names.

TileDBArray\$print(): Print summary of the array.

Usage:

```
TileDBArray$print()
```

TileDBArray\$clone(): The objects of this class are cloneable with this method.

Usage:

```
TileDBArray$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

TileDBArrayExp

Generate a TileDBArrayExp Object

Description

An enhanced version of [TileDBArray](#) with additional methods to operate on the array.

Initialization:

A new TileDBArrayExp instance is initialised using the `new()` method. Alternatively use `tdb_array()` to create a new instance and open the array at READ mode.

```
# uri path
uri <- tempdir()
# new instance
obj <- TileDBArrayExp$new(uri = uri)
# does array exist at this uri
obj$exists() # FALSE

unlink(uri)
```

Value

An object of class TileDBArrayExp, R6.

Super classes

[TileDBObject](#) -> [TileDBArray](#) -> [TileDBArrayExp](#)

Active bindings

`fragments_object` Access the [TileDBFragments](#) instance for this array.

`schema_version` Retrieve the schema version for this array.

`is_sparse` Check array schema for sparsity.

`size` Directory size.

Methods**Public methods:**

- [TileDBArrayExp\\$create\(\)](#)
- [TileDBArrayExp\\$delete_array\(\)](#)
- [TileDBArrayExp\\$reopen\(\)](#)
- [TileDBArrayExp\\$any_enums\(\)](#)
- [TileDBArrayExp\\$enum_columns\(\)](#)
- [TileDBArrayExp\\$enum_levels\(\)](#)
- [TileDBArrayExp\\$has_enumeration\(\)](#)
- [TileDBArrayExp\\$consolidate\(\)](#)
- [TileDBArrayExp\\$consolidate_async\(\)](#)
- [TileDBArrayExp\\$vacuum\(\)](#)
- [TileDBArrayExp\\$vacuum_async\(\)](#)
- [TileDBArrayExp\\$consolidate_and_vacuum\(\)](#)
- [TileDBArrayExp\\$consolidate_and_vacuum_async\(\)](#)
- [TileDBArrayExp\\$drop_attribute\(\)](#)
- [TileDBArrayExp\\$frag_num\(\)](#)
- [TileDBArrayExp\\$frag_to_vacuum\(\)](#)
- [TileDBArrayExp\\$frag_dump\(\)](#)
- [TileDBArrayExp\\$frag_uris\(\)](#)
- [TileDBArrayExp\\$schema_upgrade\(\)](#)
- [TileDBArrayExp\\$dir_tree\(\)](#)
- [TileDBArrayExp\\$clone\(\)](#)

[TileDBArrayExp\\$create\(\)](#): Create an empty Array with user defined schema.

The array will be opened at the given mode and kept opened; it can be accessed via active field `$object`.

Usage:

```
TileDBArrayExp$create(sch, mode = "WRITE")
```

Arguments:

`sch` A TileDB schema. See constructor [tiledb::tiledb_array_schema\(\)](#).

`mode` Mode to open: either "READ" or "WRITE" (default).

Returns: The object, invisibly.

TileDBArrayExp\$delete_array(): Delete the array.

Usage:

```
TileDBArrayExp$delete_array()
```

Returns: The object, invisibly.

TileDBArrayExp\$reopen(): Close and reopen the TileDB object in a new mode.

Usage:

```
TileDBArrayExp$reopen(mode = c("READ", "WRITE"))
```

Arguments:

mode New mode to open the object in; choose from: "READ" or "WRITE".

Returns: The object, invisibly.

TileDBArrayExp\$any_enums(): Checks array for factors (enumerations).

Usage:

```
TileDBArrayExp$any_enums()
```

Returns: A boolean. TRUE indicating the array has factors and FALSE otherwise.

TileDBArrayExp\$enum_columns(): Retrieve factor columns (attributes).

Usage:

```
TileDBArrayExp$enum_columns()
```

Returns: A character vector with factor columns (enumeration attributes).

TileDBArrayExp\$enum_levels(): Retrieve factor levels for a given attribute.

Usage:

```
TileDBArrayExp$enum_levels(x)
```

Arguments:

x An attribute name.

Returns: A character vector with levels (enumeration values).

TileDBArrayExp\$has_enumeration(): Check columns for factors.

Usage:

```
TileDBArrayExp$has_enumeration()
```

Returns: A logical vector indicating which column (attribute) is encoded as factor (enumeration).

TileDBArrayExp\$consolidate(): Consolidates the fragments of the array into a single fragment.

Usage:

```
TileDBArrayExp$consolidate(mode = c("fragments", "commits", "fragment_meta",  
  "array_meta"), cfg = NULL, start_time = NULL, end_time = NULL)
```

Arguments:

mode The consolidate mode, one of the following:

- "fragments": - consolidate all fragments (default)
- "commits": - consolidate all commit files
- "fragment_meta": - consolidate only fragment metadata footers to a single file
- "array_meta": - consolidate array metadata only

cfg A configuration object `tiledb::tiledb_config()` to override context configuration except for parameters set by mode, start_time, end_time. When NULL (default) the configuration parameters will be retrieved from object's context.

start_time, end_time Optional time stamp values. A date time objects of class POSIXt. If not provided, the default values from configuration object will be used.

Returns: TRUE, invisibly.

`TileDBArrayExp$consolidate_async()`: Consolidate fragments of the array into a single fragment asynchronously.

The consolidation will run in a separate R process in a clean environment.

Note this function requires the `mirai` package.

Usage:

```
TileDBArrayExp$consolidate_async(mode = c("fragments", "commits",
  "fragment_meta", "array_meta"), cfg = NULL, start_time = NULL,
  end_time = NULL)
```

Arguments:

mode The consolidate mode, one of the following:

- "fragments": - consolidate all fragments (default)
- "commits": - consolidate all commit files
- "fragment_meta": - consolidate only fragment metadata footers to a single file
- "array_meta": - consolidate array metadata only

cfg A configuration object `tiledb::tiledb_config()` to override context configuration except for parameters set by mode, start_time, end_time. When NULL (default) the configuration parameters will be retrieved from object's context.

start_time, end_time Optional time stamp values. A date time objects of class POSIXt. If not provided, the default values from configuration object will be used.

Returns: This function will return a `mirai::mirai()` object immediately. When it is resolved, it returns TRUE indicating consolidation success.

`TileDBArrayExp$vacuum()`: Clean up consolidated fragments and array metadata.

Usage:

```
TileDBArrayExp$vacuum(mode = c("fragments", "commits", "fragment_meta",
  "array_meta"), cfg = NULL)
```

Arguments:

mode The vacuum mode, one of the following:

- "fragments": - vacuum all fragments (default)
- "commits": - vacuum all commit files

- "fragment_meta": - vacuum only fragment metadata footers to a single file
- "array_meta": - vacuum array metadata only

cfg A configuration object `tiledb::tiledb_config()` to override context configuration. When NULL (default) the configuration parameters will be retrieved from object's context.

Returns: TRUE, invisibly.

`TileDBArrayExp$vacuum_async()`: Asynchronously clean up consolidated fragments and array metadata.

The clean up will run in a separate R process in a clean environment.

Note this function requires the `mirai` package.

Usage:

```
TileDBArrayExp$vacuum_async(mode = c("fragments", "commits", "fragment_meta",
  "array_meta"), cfg = NULL)
```

Arguments:

mode The vacuum mode, one of the following:

- "fragments": - vacuum all fragments (default)
- "commits": - vacuum all commit files
- "fragment_meta": - vacuum only fragment metadata footers to a single file
- "array_meta": - vacuum array metadata only

cfg A configuration object `tiledb::tiledb_config()` to override context configuration. When NULL (default) the configuration parameters will be retrieved from object's context.

Returns: This function will return a `mirai::mirai()` object immediately. When it is resolved, it returns TRUE indicating vacuum success.

`TileDBArrayExp$consolidate_and_vacuum()`: Consolidates and vacuums the fragments.

Usage:

```
TileDBArrayExp$consolidate_and_vacuum(mode = c("fragments", "commits",
  "fragment_meta", "array_meta"), cfg = NULL, start_time = NULL,
  end_time = NULL)
```

Arguments:

mode The consolidate and vacuum mode, one of the following:

- "fragments": - consolidate all fragments (default)
- "commits": - consolidate all commit files
- "fragment_meta": - consolidate only fragment metadata footers to a single file
- "array_meta": - consolidate array metadata only

cfg A configuration object `tiledb::tiledb_config()` to override context configuration except for parameters set by mode, start_time, end_time. When NULL (default) the configuration parameters will be retrieved from object's context.

start_time, end_time Optional time stamp values. A date time objects of class POSIXt. If not provided, the default values from configuration object will be used.

Returns: TRUE, invisibly.

`TileDBArrayExp$consolidate_and_vacuum_async()`: Consolidate and vacuum fragments asynchronously.

The consolidation and vacuum will run in a separate R process in a clean environment.

Note this function requires the `mirai` package.

Usage:

```
TileDBArrayExp$consolidate_and_vacuum_async(mode = c("fragments", "commits",
  "fragment_meta", "array_meta"), cfg = NULL, start_time = NULL,
  end_time = NULL)
```

Arguments:

`mode` The consolidate and vacuum mode, one of the following:

- "fragments": - consolidate all fragments (default)
- "commits": - consolidate all commit files
- "fragment_meta": - consolidate only fragment metadata footers to a single file
- "array_meta": - consolidate array metadata only

`cfg` A configuration object `tiledb::tiledb_config()` to override context configuration except for parameters set by `mode`, `start_time`, `end_time`. When `NULL` (default) the configuration parameters will be retrieved from object's context.

`start_time`, `end_time` Optional time stamp values. A date time objects of class `POSIXt`. If not provided, the default values from configuration object will be used.

Returns: This function will return a `mirai::mirai()` object immediately. When it is resolved, it returns `TRUE` indicating consolidation success.

`TileDBArrayExp$drop_attribute()`: Remove an attribute from array.

Usage:

```
TileDBArrayExp$drop_attribute(x)
```

Arguments:

`x` An attribute name.

Returns: The object, invisibly.

`TileDBArrayExp$frag_num()`: The number of fragments.

Usage:

```
TileDBArrayExp$frag_num()
```

`TileDBArrayExp$frag_to_vacuum()`: Consolidated fragments to be removed.

Usage:

```
TileDBArrayExp$frag_to_vacuum(trunc_uri = TRUE)
```

Arguments:

`trunc_uri` `TRUE` to truncate uri path.

Returns: An object of class `data.frame` with four columns:

- `Fragment`: the fragment index (starts at 1)
- `start_timestamp`: fragment's start time stamp
- `end_timestamp`: fragment's end time stamp

- URI: fragment's truncated uri path (fragment name) when `trunc_uri = TRUE` (default), otherwise the full uri path

Note that the return object will be of class `data.table` if the package is found in your system.

`TileDBArrayExp$frag_dump()`: Dump to console the commit fragments.

Usage:

```
TileDBArrayExp$frag_dump()
```

`TileDBArrayExp$frag_uris()`: Return a `data.frame` with time stamps and fragments uris.

Usage:

```
TileDBArrayExp$frag_uris(trunc_uri = TRUE)
```

Arguments:

`trunc_uri` TRUE to truncate uri path.

Returns: An object of class `data.frame` with four columns:

- `Fragment`: the fragment index (start at 1)
- `start_timestamp`: start time-stamp of when fragment was written
- `end_timestamp`: end time-stamp of when fragment was written
- `URI`: fragment's truncated uri path (fragment name) when `trunc_uri = TRUE` (default), otherwise the full uri path

Note that the return object will be of class `data.table` if the package is found in your system.

`TileDBArrayExp$schema_upgrade()`: Upgrade the array to the latest format version.

Usage:

```
TileDBArrayExp$schema_upgrade(cfg = NULL, ctx = NULL)
```

Arguments:

`cfg` A configuration object `tiledb::tiledb_config()`.

`ctx` Optional `tiledb::tiledb_ctx()` object. By default, object's context is used.

`TileDBArrayExp$dir_tree()`: Print directory contents.

Usage:

```
TileDBArrayExp$dir_tree(recursive = TRUE)
```

Arguments:

`recursive` Should it recurse fully? Defaults to TRUE.

Returns: A character vector with file paths, invisibly.

`TileDBArrayExp$clone()`: The objects of this class are cloneable with this method.

Usage:

```
TileDBArrayExp$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

TileDBFragments

Generate a TileDBFragments Object

Description

A class for working with TileDB Fragments.

Initialization:

A new TileDBFragments instance is initialised using the `new()` method or with the functional interface `tdb_fragments()`.

Value

An object of class TileDBFragments, R6.

Active bindings

`uri` The URI of the TileDB object.

`fragment_info` Get the TileDB Fragment Info object as returned by `tiledb::tiledb_fragment_info`.

Methods

Public methods:

- `TileDBFragments$new()`
- `TileDBFragments$class()`
- `TileDBFragments$frag_num()`
- `TileDBFragments$frag_uris()`
- `TileDBFragments$reload_finfo()`
- `TileDBFragments$to_vacuum()`
- `TileDBFragments$to_vacuum_num()`
- `TileDBFragments$delete_fragment_range()`
- `TileDBFragments$delete_fragment_list()`
- `TileDBFragments$delete_fragment()`
- `TileDBFragments$get_ifragment()`
- `TileDBFragments$get_first_ifragments()`
- `TileDBFragments$get_last_ifragments()`
- `TileDBFragments$dump()`
- `TileDBFragments$print()`

`TileDBFragments$new()`: Create a new TileDBFragments instance.

Usage:

```
TileDBFragments$new(uri, ctx = NULL)
```

Arguments:

`uri` URI path for the TileDB Array.

ctx Optional `tiledb::tiledb_ctx()` object.

`TileDBFragments$class()`: Print the name of the R6 class.

Usage:

```
TileDBFragments$class()
```

`TileDBFragments$frag_num()`: Get the number of fragments.

Usage:

```
TileDBFragments$frag_num()
```

Returns: A numeric value.

`TileDBFragments$frag_uris()`: Return a `data.frame` with time stamps and fragments uris.

Usage:

```
TileDBFragments$frag_uris(trunc_uri = TRUE)
```

Arguments:

`trunc_uri` TRUE for fragment name in the form: `__ts1_ts2_<label>_<ver>`, otherwise FALSE for full path.

Returns: An object of class `data.frame` with four columns:

- `Fragment`: the fragment index (start at 1)
- `start_timestamp`: start time-stamp of when fragment was written
- `end_timestamp`: end time-stamp of when fragment was written
- `URI`: fragment's truncated uri path (fragment name) when `trunc_uri = TRUE` (default), otherwise the full uri path

Note that the return object will be of class `data.table` if the package is found in your system.

`TileDBFragments$reload_finfo()`: Refresh the TileDB Fragment Info object.

Usage:

```
TileDBFragments$reload_finfo()
```

Returns: The object, invisibly.

`TileDBFragments$to_vacuum()`: Consolidated fragments to be removed.

Usage:

```
TileDBFragments$to_vacuum(trunc_uri = TRUE)
```

Arguments:

`trunc_uri` TRUE for fragment name in the form: `__ts1_ts2_<label>_<ver>`, otherwise FALSE for full path.

Returns: An object of class `data.frame` with four columns:

- `Fragment`: the fragment index (starts at 1)
- `start_timestamp`: fragment's start time stamp
- `end_timestamp`: fragment's end time stamp
- `URI`: fragment's truncated uri path (fragment name) when `trunc_uri = TRUE` (default), otherwise the full uri path

Note that the return object will be of class `data.table` if the package is found in your system.

TileDBFragments\$to_vacuum_num(): Return the number of fragments to vacuum

Usage:

TileDBFragments\$to_vacuum_num()

Returns: A numeric value.

TileDBFragments\$delete_fragment_range(): Delete fragments using a time-stamp range.

Usage:

TileDBFragments\$delete_fragment_range(start_time, end_time)

Arguments:

start_time, end_time Time stamp values. A date time objects of class POSIXct.

Returns: A logical TRUE, invisibly. Note that if time stamps fall outside fragments' time range no deletion will incur.

TileDBFragments\$delete_fragment_list(): Delete fragments using a vector of fragment uris.

Use \$frag_uris(trunc_uri = FALSE) method to get a data.frame with all fragment uri paths.

Usage:

TileDBFragments\$delete_fragment_list(frag_uris)

Arguments:

frag_uris A vector of fragment uris.

Returns: TRUE for successful deletion, invisibly.

TileDBFragments\$delete_fragment(): Delete a fragment by index.

Usage:

TileDBFragments\$delete_fragment(n)

Arguments:

n A fragment index to be deleted (starts at 1).

Returns: A boolean, invisibly. TRUE for successful deletion and FALSE for no fragment to delete.

TileDBFragments\$get_ifragment(): Get a fragment by index.

Usage:

TileDBFragments\$get_ifragment(fid)

Arguments:

fid A fragment index (starts at 1).

Returns: An object of class ifragment.

TileDBFragments\$get_first_ifragments(): Get the first n fragments.

Usage:

TileDBFragments\$get_first_ifragments(n)

Arguments:

n A numeric value.

Returns: An object of class ifragment_list.

TileDBFragments\$get_last_ifragments(): Get the last n fragments.

Usage:

```
TileDBFragments$get_last_ifragments(n)
```

Arguments:

n A numeric value.

Returns: An object of class ifragment_list.

TileDBFragments\$dump(): Dump to console the commit fragments.

Usage:

```
TileDBFragments$dump()
```

TileDBFragments\$print(): Print Fragments class.

Usage:

```
TileDBFragments$print()
```

TileDBGroup

Generate a TileDBGroup Object

Description

Base class for representing a TileDB Group.

Initialization:

A new TileDBGroup instance is initialized using the new() method:

```
# uri path
uri <- tempdir()
# new instance
obj <- TileDBGroup$new(uri = uri)
# does array exist at this uri
obj$exists() # FALSE

unlink(uri)
```

Value

An object of class TileDBGroup, R6.

Super class

[TileDBObject](#) -> TileDBGroup

Active bindings

`object` Access the underlying `tiledb::tiledb_group()` object.

`members` Access the list of group members.

If TileDB group object is closed, it will be opened to get members and close on exit.

Methods**Public methods:**

- `TileDBGroup$create()`
- `TileDBGroup$open()`
- `TileDBGroup$close()`
- `TileDBGroup$remove()`
- `TileDBGroup$delete()`
- `TileDBGroup$count_members()`
- `TileDBGroup$get_members_df()`
- `TileDBGroup$get_member()`
- `TileDBGroup$set_member()`
- `TileDBGroup$names()`
- `TileDBGroup$member_exists()`
- `TileDBGroup$print()`
- `TileDBGroup$dump()`
- `TileDBGroup$clone()`

`TileDBGroup$create()`: Create a TileDB Group object at the given URI path.

Usage:

```
TileDBGroup$create(mode = "WRITE")
```

Arguments:

`mode` Mode to open : either "READ" or "WRITE" (default).

Returns: The object, invisibly.

`TileDBGroup$open()`: Open TileDB group object in read or write mode.

Usage:

```
TileDBGroup$open(mode = c("READ", "WRITE"))
```

Arguments:

`mode` Mode to open : either "READ" or "WRITE". Default is "READ".

Returns: The object, invisibly.

`TileDBGroup$close()`: Close the group object.

All instantiated group members will be closed if opened, and before closing the group object.

Usage:

```
TileDBGroup$close()
```

Returns: The object, invisibly.

TileDBGroup\$remove(): Remove member.

Removes an array or group resource from TileDBGroup member list.

Usage:

```
TileDBGroup$remove(name)
```

Arguments:

name Name of the member to remove.

Returns: The object, invisibly.

TileDBGroup\$delete(): Delete member.

Deletes a TileDBGroup's member from disk and removes it from member list.

Usage:

```
TileDBGroup$delete(name)
```

Arguments:

name Name of the member to delete.

Returns: The object, invisibly.

TileDBGroup\$count_members(): Count the number of members in the group.

Usage:

```
TileDBGroup$count_members()
```

Returns: The number of members in the group.

TileDBGroup\$get_members_df(): List group members.

Usage:

```
TileDBGroup$get_members_df(type = c("ALL", "GROUP", "ARRAY"))
```

Arguments:

type Select type member, either "ALL", "GROUP" or "ARRAY". By default all member types are listed.

Returns: A data.frame with columns name, type, and uri.

TileDBGroup\$get_member(): Retrieve a group member by name. If the member isn't already open, it is opened in the same mode as the parent.

Usage:

```
TileDBGroup$get_member(name)
```

Arguments:

name The name of the member.

Returns: A TileDBArray or TileDBGroup.

TileDBGroup\$set_member(): Add new member to the group.

Usage:

```
TileDBGroup$set_member(object, name = NULL, relative = NULL)
```

Arguments:

object A TileDBArray or TileDBGroup object to add.
name Name to use for the member. By default the base name of the object's URI is used.
relative An optional logical value indicating whether the new object's URI is relative to the group's URI. If NULL (the default), the object's URI is assumed to be relative unless it is a tiledb:// URI.

Returns: The object, invisibly.

TileDBGroup\$names(): Retrieve the members' names.

Usage:

```
TileDBGroup$names()
```

Returns: A character vector of member names.

TileDBGroup\$member_exists(): Check if a member exists.

Usage:

```
TileDBGroup$member_exists(name)
```

Arguments:

name Name of the member to check.

Returns: A logical value.

TileDBGroup\$print(): Print summary of the group.

Usage:

```
TileDBGroup$print()
```

TileDBGroup\$dump(): Dump the TileDB Group structure to string.

Usage:

```
TileDBGroup$dump(title = "TileDB Directory", recursive = TRUE)
```

Arguments:

title A character string for title header. Set NULL to omit.

recursive Should the nested uris be returned recursively? Default is TRUE.

Returns: A character string, invisibly.

TileDBGroup\$clone(): The objects of this class are cloneable with this method.

Usage:

```
TileDBGroup$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

TileDBGroupExp	<i>Generate a TileDBGroupExp Object</i>
----------------	---

Description

An enhanced version of [TileDBGroup](#) with additional methods to operate on the group.

Initialization:

A new TileDBGroupExp instance is initialised using the `new()` method. Alternatively use `tdb_group()` to create a new instance and open the group at READ mode.

```
# uri path
uri <- tempdir()
# new instance
obj <- TileDBGroupExp$new(uri = uri)
# does group exist at this uri
obj$exists() # FALSE

unlink(uri)
```

Value

An object of class TileDBGroupExp, R6.

Super classes

[TileDBObject](#) -> [TileDBGroup](#) -> TileDBGroupExp

Active bindings

`size` Directory size.

Methods

Public methods:

- [TileDBGroupExp\\$has_non_members\(\)](#)
- [TileDBGroupExp\\$non_members\(\)](#)
- [TileDBGroupExp\\$prune_non_members\(\)](#)
- [TileDBGroupExp\\$delete_group\(\)](#)
- [TileDBGroupExp\\$walk_group\(\)](#)
- [TileDBGroupExp\\$dir_tree\(\)](#)
- [TileDBGroupExp\\$clone\(\)](#)

`TileDBGroupExp$has_non_members()`: Checks for non group members at group's uri path. This function compares the TileDB resources at group's uri path to its members to determine if there are any non member.

Usage:

```
TileDBGroupExp$has_non_members()
```

Returns: TRUE for having non members, FALSE otherwise.

TileDBGroupExp\$non_members(): List TileDB resources which are not members at group's uri path.

This function compares the TileDB resources at group's uri path to its members and returns a `data.frame` with non members object type and uri path.

Usage:

```
TileDBGroupExp$non_members()
```

Returns: An object of class `data.frame` with columns TYPE and URI of non member TileDB resources.

TileDBGroupExp\$prune_non_members(): Delete TileDB resources which are not members.

Usage:

```
TileDBGroupExp$prune_non_members()
```

Returns: An character vector of deleted uri paths.

TileDBGroupExp\$delete_group(): Delete written data from Group.

This function deletes all written data from a `tiledb_group` object, i.e., the folder will not be consider as a TileDB Group after the operation. Any other data will not be deleted unless we set `recursive = TRUE`.

Usage:

```
TileDBGroupExp$delete_group(recursive = FALSE)
```

Arguments:

`recursive` Should all data be deleted inside the group? Default is FALSE.

Returns: The object, invisibly.

TileDBGroupExp\$walk_group(): List all TileDB resources at group's uri path.

Usage:

```
TileDBGroupExp$walk_group(order = c("pre", "post"))
```

Arguments:

`order` Traversal order, either pre (default) or post.

Returns: An object of class `data.frame` with columns TYPE and URI with all TileDB resources under group's uri path.

TileDBGroupExp\$dir_tree(): Print directory contents.

Usage:

```
TileDBGroupExp$dir_tree(recursive = TRUE)
```

Arguments:

`recursive` Should it recurse fully? Defaults to TRUE.

Returns: A character vector with file paths, invisibly.

TileDBGroupExp\$clone(): The objects of this class are cloneable with this method.

Usage:

```
TileDBGroupExp$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

vfs_dir_tree

Print Directory Contents

Description

A modified version of `fs::dir_tree()` to work with TileDB VFS.

Usage

```
vfs_dir_tree(uri, recursive = TRUE, vfs = NULL)
```

Arguments

uri	URI path for the TileDB object.
recursive	Should it recurse fully? Defaults to TRUE.
vfs	A <code>tiledb::tiledb_vfs()</code> object. If NULL (default) will create a new VFS object.

Value

A character vector with file paths, invisibly.

Examples

```
## Not run:
# URI path
uri <- tempfile()

# Demo array
demo_array_UCBAdmissions(uri)

# Array instance
arobj <- tdb_array(uri)

# Print directory contents
arobj$dir_tree()

## End(Not run)
```

vfs_size	<i>VFS Directory Size</i>
----------	---------------------------

Description

VFS Directory Size

Usage

```
vfs_size(uri, vfs = NULL)
```

Arguments

uri	URI path for the TileDB object.
vfs	A <code>tiledb::tiledb_vfs()</code> object. If NULL (default) will create a new VFS object.

Value

The size of the directory.

Examples

```
## Not run:  
# URI path  
uri <- tempfile()  
  
# Demo array  
demo_UCBAdmissions_array(uri)  
  
# Directory size  
vfs_size(uri)  
  
## End(Not run)
```

Index

array_timestamps, [2](#)
array_timestamps(), [7](#)
as.POSIXct(), [14](#)
attr(), [8](#)

close, [3](#)

delete_metadata, [4](#)
demo_array_UCBAdmissions, [5](#)

fetch_metadata, [5](#)
fetch_metadata(), [8](#), [14](#)
fs::dir_tree(), [39](#)

group_timestamps, [6](#)
group_timestamps(), [3](#)

metadata, [7](#)
metadata(), [4](#), [6](#), [14](#)
metadata<-character(metadata), [7](#)
metadata<-tiledb_array(metadata), [7](#)
metadata<-tiledb_group(metadata), [7](#)
metadata<-TileDBArray(metadata), [7](#)
metadata<-TileDBGroup(metadata), [7](#)
mirai::mirai(), [26–28](#)

open_write, [9](#)

set_config_params, [10](#)
set_config_params(), [11](#), [12](#)
set_consolidation_timestamps, [11](#)
set_group_timestamps, [12](#)
set_metadata, [13](#)
set_metadata(), [4](#), [6](#), [8](#)
set_tiledb_timestamp, [14](#)
set_tiledb_timestamp(), [3](#), [7](#), [15](#), [19](#)
Sys.timezone(), [3](#), [7](#), [14](#)

tdb_array, [15](#)
tdb_array(), [23](#)
tdb_array_create, [16](#)

tdb_fragments, [17](#)
tdb_fragments(), [30](#)
tdb_group, [18](#)
tdb_group_create, [20](#)
tiledb::tiledb_array, [22](#)
tiledb::tiledb_array(), [3](#), [9](#), [15](#), [21](#)
tiledb::tiledb_array_schema, [22](#)
tiledb::tiledb_array_schema(), [17](#), [24](#)
tiledb::tiledb_attr, [22](#)
tiledb::tiledb_config(), [10–12](#), [26–29](#)
tiledb::tiledb_ctx(), [4](#), [6](#), [8](#), [9](#), [13](#), [15](#),
[17–20](#), [29](#), [31](#)
tiledb::tiledb_dim, [22](#)
tiledb::tiledb_fragment_info, [18](#), [30](#)
tiledb::tiledb_group(), [3](#), [9](#), [19](#), [34](#)
tiledb::tiledb_vfs(), [39](#), [40](#)
TileDBArray, [3](#), [9](#), [20](#), [23](#), [24](#)
TileDBArrayExp, [15](#), [23](#)
TileDBFragments, [16](#), [17](#), [24](#), [30](#)
TileDBGroup, [3](#), [9](#), [33](#), [37](#)
TileDBGroupExp, [18](#), [37](#)
TileDBObject, [14](#), [21](#), [24](#), [33](#), [37](#)

unset_config_params
 (set_config_params), [10](#)
unset_consolidation_timestamps
 (set_consolidation_timestamps),
 [11](#)
unset_group_timestamps
 (set_group_timestamps), [12](#)

vfs_dir_tree, [39](#)
vfs_size, [40](#)