

# Package ‘tiledb’

October 8, 2020

**Type** Package

**Version** 0.8.1

**Title** Sparse and Dense Multidimensional Array Storage Engine for Data Science

**Author** TileDB, Inc.

**Copyright** TileDB, Inc.

**Maintainer** Dirk Eddelbuettel <dirk@tiledb.com>

**Description** The data science storage engine 'TileDB' introduces a powerful on-disk format for multi-dimensional arrays. It supports dense and sparse arrays, dataframes and key-values stores, cloud storage ('S3', 'GCS', 'Azure'), chunked arrays, multiple compression, encryption and checksum filters, uses a fully multi-threaded implementation, supports parallel I/O, data versioning ('time travel'), metadata and groups. It is implemented as an embeddable cross-platform C++ library with APIs from several languages, and integrations.

**License** MIT + file LICENSE

**URL** <https://github.com/TileDB-Inc/TileDB-R>

**BugReports** <https://github.com/TileDB-Inc/TileDB-R/issues>

**SystemRequirements** cmake (only when TileDB source build selected), git (only when TileDB source build selected); on x86\_64 platforms pre-built TileDB Embedded libraries are available at GitHub and are used if no TileDB installation is detected, and no other option to build or download was specified by the user.

**Imports** methods, Rcpp, nanotime

**LinkingTo** Rcpp

**Suggests** tinytest, rmarkdown, knitr, BiocStyle, curl, bit64

**VignetteBuilder** knitr

**RoxygenNote** 7.1.1

**Encoding** UTF-8

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2020-10-08 04:10:02 UTC

## R topics documented:

tiledb-package . . . . .	5
allows_dups . . . . .	6
allows_dups<- . . . . .	6
as.data.frame.tiledb_config . . . . .	7
as.vector.tiledb_config . . . . .	7
as_data_frame . . . . .	8
attrs,tiledb_array,ANY-method . . . . .	9
attrs,tiledb_array_schema,ANY-method . . . . .	9
attrs,tiledb_array_schema,character-method . . . . .	10
attrs,tiledb_array_schema,numeric-method . . . . .	11
attrs,tiledb_dense,ANY-method . . . . .	11
attrs,tiledb_sparse,ANY-method . . . . .	12
attrs<- . . . . .	12
attrs<-,tiledb_array-method . . . . .	13
attrs<-,tiledb_sparse-method . . . . .	13
cell_order,tiledb_array_schema-method . . . . .	14
cell_val_num,tiledb_attr-method . . . . .	14
config,tiledb_ctx-method . . . . .	15
datatype,tiledb_attr-method . . . . .	15
datatype,tiledb_dim-method . . . . .	16
datatype,tiledb_domain-method . . . . .	17
dim.tiledb_array_schema . . . . .	17
dim.tiledb_dim . . . . .	18
dim.tiledb_domain . . . . .	19
dimensions,tiledb_array_schema-method . . . . .	19
dimensions,tiledb_domain-method . . . . .	20
domain . . . . .	21
domain,tiledb_array_schema-method . . . . .	22
domain,tiledb_dim-method . . . . .	22
extended . . . . .	23
extended<- . . . . .	23
filter_list,tiledb_array_schema-method . . . . .	24
filter_list,tiledb_attr-method . . . . .	24
fromDataFrame . . . . .	25
is.anonymous . . . . .	26
is.anonymous.tiledb_dim . . . . .	27
is.integral,tiledb_domain-method . . . . .	27
is.sparse,tiledb_array_schema-method . . . . .	28
is.sparse,tiledb_dense-method . . . . .	28
is.sparse,tiledb_sparse-method . . . . .	29
limitTileDBCores . . . . .	29

max_chunk_size,tiledb_filter_list-method . . . . .	30
name,tiledb_attr-method . . . . .	31
name,tiledb_dim-method . . . . .	31
nfilters,tiledb_filter_list-method . . . . .	32
print.tiledb_metadata . . . . .	33
return.data.frame . . . . .	33
return.data.frame,tiledb_array-method . . . . .	34
return.data.frame,tiledb_sparse-method . . . . .	34
return.data.frame<- . . . . .	35
return.data.frame<-,tiledb_array-method . . . . .	35
return.data.frame<-,tiledb_sparse-method . . . . .	36
r_to_tiledb_type . . . . .	36
schema,tiledb_array-method . . . . .	37
schema,tiledb_dense-method . . . . .	37
schema,tiledb_sparse-method . . . . .	38
selected_ranges . . . . .	38
selected_ranges<- . . . . .	39
set_max_chunk_size,tiledb_filter_list,numeric-method . . . . .	39
show,tiledb_array-method . . . . .	40
show,tiledb_array_schema-method . . . . .	40
show,tiledb_attr-method . . . . .	41
show,tiledb_config-method . . . . .	41
show,tiledb_dense-method . . . . .	42
show,tiledb_domain-method . . . . .	42
show,tiledb_sparse-method . . . . .	43
tile,tiledb_dim-method . . . . .	43
tiledb_array . . . . .	44
tiledb_array-class . . . . .	45
tiledb_array_close . . . . .	45
tiledb_array_create . . . . .	46
tiledb_array_is_heterogeneous . . . . .	46
tiledb_array_is_homogeneous . . . . .	47
tiledb_array_open . . . . .	47
tiledb_array_schema . . . . .	48
tiledb_array_schema-class . . . . .	49
tiledb_attr . . . . .	49
tiledb_attr-class . . . . .	50
tiledb_config . . . . .	50
tiledb_config-class . . . . .	51
tiledb_config_load . . . . .	51
tiledb_config_save . . . . .	52
tiledb_ctx . . . . .	52
tiledb_ctx-class . . . . .	53
tiledb_ctx_set_default_tags . . . . .	53
tiledb_ctx_set_tag . . . . .	54
tiledb_delete_metadata . . . . .	54
tiledb_dense . . . . .	55
tiledb_dense-class . . . . .	55

tiledb_dim . . . . .	56
tiledb_dim-class . . . . .	57
tiledb_domain . . . . .	57
tiledb_domain-class . . . . .	57
tiledb_filter . . . . .	58
tiledb_filter-class . . . . .	59
tiledb_filter_get_option . . . . .	59
tiledb_filter_list . . . . .	60
tiledb_filter_list-class . . . . .	60
tiledb_filter_set_option . . . . .	61
tiledb_filter_type . . . . .	61
tiledb_get_all_metadata . . . . .	62
tiledb_get_context . . . . .	62
tiledb_get_metadata . . . . .	63
tiledb_group_create . . . . .	63
tiledb_has_metadata . . . . .	64
tiledb_is_supported_fs . . . . .	64
tiledb_ndim,tiledb_array_schema-method . . . . .	65
tiledb_ndim,tiledb_dim-method . . . . .	66
tiledb_ndim,tiledb_domain-method . . . . .	66
tiledb_num_metadata . . . . .	67
tiledb_object_ls . . . . .	67
tiledb_object_mv . . . . .	68
tiledb_object_rm . . . . .	68
tiledb_object_type . . . . .	69
tiledb_object_walk . . . . .	69
tiledb_put_metadata . . . . .	70
tiledb_query . . . . .	70
tiledb_query-class . . . . .	71
tiledb_query_add_range . . . . .	71
tiledb_query_add_range_with_type . . . . .	72
tiledb_query_alloc_buffer_ptr_char . . . . .	72
tiledb_query_alloc_buffer_ptr_char_subarray . . . . .	73
tiledb_query_buffer_alloc_ptr . . . . .	74
tiledb_query_create_buffer_ptr . . . . .	74
tiledb_query_create_buffer_ptr_char . . . . .	75
tiledb_query_finalize . . . . .	75
tiledb_query_get_buffer_char . . . . .	76
tiledb_query_get_buffer_ptr . . . . .	76
tiledb_query_get_layout . . . . .	77
tiledb_query_result_buffer_elements . . . . .	77
tiledb_query_set_buffer . . . . .	78
tiledb_query_set_buffer_ptr . . . . .	78
tiledb_query_set_buffer_ptr_char . . . . .	79
tiledb_query_set_layout . . . . .	79
tiledb_query_set_subarray . . . . .	80
tiledb_query_status . . . . .	80
tiledb_query_submit . . . . .	81

tiledb_query_type . . . . .	81
tiledb_schema_get_names . . . . .	82
tiledb_schema_get_types . . . . .	82
tiledb_set_context . . . . .	83
tiledb_sparse . . . . .	83
tiledb_sparse-class . . . . .	84
tiledb_stats_disable . . . . .	84
tiledb_stats_dump . . . . .	85
tiledb_stats_enable . . . . .	85
tiledb_subarray . . . . .	86
tiledb_version . . . . .	86
tiledb_vfs . . . . .	87
tiledb_vfs-class . . . . .	87
tiledb_vfs_create_bucket . . . . .	88
tiledb_vfs_create_dir . . . . .	88
tiledb_vfs_empty_bucket . . . . .	89
tiledb_vfs_file_size . . . . .	89
tiledb_vfs_is_bucket . . . . .	90
tiledb_vfs_is_dir . . . . .	90
tiledb_vfs_is_empty_bucket . . . . .	91
tiledb_vfs_is_file . . . . .	92
tiledb_vfs_move_dir . . . . .	92
tiledb_vfs_move_file . . . . .	93
tiledb_vfs_remove_bucket . . . . .	93
tiledb_vfs_remove_dir . . . . .	94
tiledb_vfs_remove_file . . . . .	94
tiledb_vfs_touch . . . . .	95
tile_order,tiledb_array_schema-method . . . . .	95
[,tiledb_array,ANY-method . . . . .	96
[,tiledb_config,ANY-method . . . . .	96
[,tiledb_dense,ANY-method . . . . .	97
[,tiledb_filter_list,ANY-method . . . . .	98
[,tiledb_sparse,ANY-method . . . . .	98
[<-,tiledb_array,ANY,ANY,ANY-method . . . . .	99
[<-,tiledb_config,ANY,ANY,ANY-method . . . . .	100
[<-,tiledb_dense,ANY,ANY,ANY-method . . . . .	101
[<-,tiledb_sparse,ANY,ANY,ANY-method . . . . .	101

<b>Index</b>	<b>103</b>
--------------	------------

---

tiledb-package	<i>tiledb - Interface to the TileDB Storage Manager API</i>
----------------	---

---

## Description

The efficient multi-dimensional array management system 'TileDB' introduces a novel on-disk format that can effectively store reads. It features excellent compression, an efficient parallel I/O system which also scales well, and bindings to multiple languages.

---

allows_dups	<i>Returns logical value whether the array schema allows duplicate values or not. This is only valid for sparse arrays.</i>
-------------	---

---

### Description

Returns logical value whether the array schema allows duplicate values or not. This is only valid for sparse arrays.

### Usage

```
allows_dups(x)

## S4 method for signature 'tiledb_array_schema'
allows_dups(x)
```

### Arguments

x	tiledb_array_schema
---	---------------------

### Value

the logical value

---

allows_dups<-	<i>Sets toggle whether the array schema allows duplicate values or not. This is only valid for sparse arrays.</i>
---------------	---

---

### Description

Sets toggle whether the array schema allows duplicate values or not. This is only valid for sparse arrays.

### Usage

```
allows_dups(x) <- value

## S4 replacement method for signature 'tiledb_array_schema'
allows_dups(x) <- value
```

### Arguments

x	tiledb_array_schema
value	logical value

**Value**

the tiledb\_array\_schema object

---

as.data.frame.tiledb\_config

*Convert a tiledb\_config object to a R data.frame*

---

**Description**

Convert a tiledb\_config object to a R data.frame

**Usage**

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

**Arguments**

x	tiledb_config object
...	Extra parameter for method signature, currently unused.

**Value**

a data.frame with parameter, value columns

**Examples**

```
cfg <- tiledb_config()  
as.data.frame(cfg)
```

---

as.vector.tiledb\_config

*Convert a tiledb\_config object to a R vector*

---

**Description**

Convert a tiledb\_config object to a R vector

**Usage**

```
## S3 method for class 'tiledb_config'  
as.vector(x, mode = "any")
```

**Arguments**

x	tiledb_config object
mode	Character value "any", currently unused

**Value**

a character vector of config parameter names, values

**Examples**

```
cfg <- tiledb_config()
as.vector(cfg)
```

---

as_data_frame	<i>Construct a data.frame from query results</i>
---------------	--

---

**Description**

Converts a tiledb object to a data.frame object

**Usage**

```
as_data_frame(dom, data, extended = FALSE)
```

**Arguments**

dom	tiledb_domain object
data	tiledb object to be converted
extended	optional logical variable selected wider display with coordinates, defaults to false

**Value**

data.frame object constructed from data



---

attrs, tiledb\_array, ANY-method

*Retrieve attributes from tiledb\_array object*

---

### Description

By default, all attributes will be selected. But if a subset of attribute names is assigned to the internal slot `attrs`, then only those attributes will be queried. This methods accesses the slot.

### Usage

```
## S4 method for signature 'tiledb_array,ANY'
attrs(object)
```

### Arguments

`object`                      A `tiledb_array` object

### Value

An empty character vector if no attributes have been selected or else a vector with attributes.

---

attrs, tiledb\_array\_schema, ANY-method

*Returns a list of all tiledb\_attr objects associated with the tiledb\_array\_schema*

---

### Description

Returns a list of all `tiledb_attr` objects associated with the `tiledb_array_schema`

### Usage

```
## S4 method for signature 'tiledb_array_schema,ANY'
attrs(object, idx, ...)
```

### Arguments

`object`                      `tiledb_array_schema`  
`idx`                          index argument, currently unused.  
`...`                        Extra parameter for method signature, currently unused.

### Value

a list of `tiledb_attr` objects

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32"),
                                          tiledb_attr("a2", type = "FLOAT64")))

attrs(sch)

lapply(attrs(sch), datatype)
```

---

attrs, tiledb\_array\_schema, character-method

*Returns a tiledb\_attr object associated with the tiledb\_array\_schema with a given name.*

---

**Description**

Returns a tiledb\_attr object associated with the tiledb\_array\_schema with a given name.

**Usage**

```
## S4 method for signature 'tiledb_array_schema,character'
attrs(object, idx, ...)
```

**Arguments**

object	tiledb_array_schema
idx	attribute name string
...	Extra parameter for method signature, currently unused.

**Value**

a tiledb\_attr object

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32"),
                                          tiledb_attr("a2", type = "FLOAT64")))

attrs(sch, "a2")
```

---

```
attrs, tiledb_array_schema, numeric-method
```

*Returns a tiledb\_attr object associated with the tiledb\_array\_schema with a given index*

---

### Description

The attribute index is defined by the order the attributes were defined in the schema

### Usage

```
## S4 method for signature 'tiledb_array_schema,numeric'
attrs(object, idx, ...)
```

### Arguments

object	tiledb_array_schema
idx	attribute index
...	Extra parameter for method signature, currently unused.

### Value

a tiledb\_attr object

### Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32"),
                                          tiledb_attr("a2", type = "FLOAT64")))
attrs(sch, 2)
```

---

```
attrs, tiledb_dense, ANY-method
```

*Retrieve attributes from tiledb\_dense object*

---

### Description

By default, all attributes will be selected. But if a subset of attribute names is assigned to the internal slot attrs, then only those attributes will be queried. This methods accesses the slot.

### Usage

```
## S4 method for signature 'tiledb_dense,ANY'
attrs(object)
```

**Arguments**

object                    A tiledb\_dense array object

**Value**

An empty character vector if no attributes have been selected or else a vector with attributes.

---

```
attrs, tiledb_sparse, ANY-method
```

*Retrieve attributes from tiledb\_sparse object*

---

**Description**

By default, all attributes will be selected. But if a subset of attribute names is assigned to the internal slot `attrs`, then only those attributes will be queried. This methods accesses the slot.

**Usage**

```
## S4 method for signature 'tiledb_sparse,ANY'
attrs(object)
```

**Arguments**

object                    A tiledb\_sparse array object

**Value**

An empty character vector if no attributes have been selected or else a vector with attributes.

---

```
attrs<-
```

*Selects attributes for the given TileDB array*

---

**Description**

Selects attributes for the given TileDB array

**Usage**

```
attrs(x) <- value

## S4 replacement method for signature 'tiledb_dense'
attrs(x) <- value
```

### Arguments

x	A tiledb_dense array object
value	A character vector with attributes

### Value

The modified tiledb\_dense array object

---

*attrs<-, tiledb\_array-method*  
*Selects attributes for the given TileDB array*

---

### Description

Selects attributes for the given TileDB array

### Usage

```
## S4 replacement method for signature 'tiledb_array'
attrs(x) <- value
```

### Arguments

x	A tiledb_array object
value	A character vector with attributes

### Value

The modified tiledb\_array object

---

*attrs<-, tiledb\_sparse-method*  
*Selects attributes for the given TileDB array*

---

### Description

Selects attributes for the given TileDB array

### Usage

```
## S4 replacement method for signature 'tiledb_sparse'
attrs(x) <- value
```

**Arguments**

- x                    A tiledb\_sparse array object
- value                A character vector with attributes

**Value**

The modified tiledb\_sparse array object

---

cell_order,tiledb_array_schema-method
<i>Returns the cell layout string associated with the tiledb_array_schema</i>

---

**Description**

Returns the cell layout string associated with the tiledb\_array\_schema

**Usage**

```
## S4 method for signature 'tiledb_array_schema'  
cell_order(object)
```

**Arguments**

- object                tiledb object

---

cell_val_num,tiledb_attr-method
<i>Return the number of scalar values per attribute cell</i>

---

**Description**

Return the number of scalar values per attribute cell

**Usage**

```
## S4 method for signature 'tiledb_attr'  
cell_val_num(object)
```

**Arguments**

- object                tiledb\_attr object

**Value**

integer number of cells

**Examples**

```
a1 <- tiledb_attr("a1", type = "FLOAT64", ncells = 1)
cell_val_num(a1)
```

---

```
config, tiledb_ctx-method
```

*Retrieve the tiledb\_config object from the tiledb\_ctx*

---

**Description**

Retrieve the tiledb\_config object from the tiledb\_ctx

**Usage**

```
## S4 method for signature 'tiledb_ctx'
config(object = tiledb_get_context())
```

**Arguments**

object            tiledb\_ctx object

**Value**

tiledb\_config object associated with the tiledb\_ctx instance

**Examples**

```
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "10"))
cfg <- config(ctx)
cfg["sm.tile_cache_size"]
```

---

```
datatype, tiledb_attr-method
```

*Return the tiledb\_attr datatype*

---

**Description**

Return the tiledb\_attr datatype

**Usage**

```
## S4 method for signature 'tiledb_attr'
datatype(object)
```

**Arguments**

object                    tiledb\_attr object

**Value**

tiledb datatype string

**Examples**

```
a1 <- tiledb_attr("a1", type = "INT32")
datatype(a1)

a2 <- tiledb_attr("a1", type = "FLOAT64")
datatype(a2)
```

---

datatype, tiledb\_dim-method

*Return the tiledb\_dim datatype*

---

**Description**

Return the tiledb\_dim datatype

**Usage**

```
## S4 method for signature 'tiledb_dim'
datatype(object)
```

**Arguments**

object                    tiledb\_dim object

**Value**

tiledb datatype string

**Examples**

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L), tile = 2L, type = "INT32")
datatype(d1)
```



---

```
datatype, tiledb_domain-method
      Returns tiledb_domain TileDB type string
```

---

**Description**

Returns tiledb\_domain TileDB type string

**Usage**

```
## S4 method for signature 'tiledb_domain'
datatype(object)
```

**Arguments**

object                    tiledb\_domain

**Value**

tiledb\_domain type string

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32")))
datatype(dom)
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
datatype(dom)
```

---

```
dim.tiledb_array_schema
      Retrieve the dimension (domain extent) of the domain
```

---

**Description**

Only valid for integral (integer) domains

**Usage**

```
## S3 method for class 'tiledb_array_schema'
dim(x)
```

**Arguments**

x                            tiledb\_array\_schema

**Value**

a dimension vector

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32"),
                                          tiledb_attr("a2", type = "FLOAT64")))
dim(sch)
```

---

dim.tiledb_dim	<i>Retrieves the dimension of the tiledb_dim domain</i>
----------------	---

---

**Description**

Retrieves the dimension of the tiledb\_dim domain

**Usage**

```
## S3 method for class 'tiledb_dim'
dim(x)
```

**Arguments**

x tiledb\_dim object

**Value**

a vector of the tile\_dim domain type, of the dim domain dimension (extent)

**Examples**

```
d1 <- tiledb_dim("d1", c(1L, 10L), 5L)
dim(d1)
```

---

dim.tiledb_domain	<i>Retrieve the dimension (domain extent) of the domain</i>
-------------------	---

---

**Description**

Only valid for integral (integer) domains

**Usage**

```
## S3 method for class 'tiledb_domain'
dim(x)
```

**Arguments**

x	tiledb_domain
---	---------------

**Value**

dimension vector

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32"),
                             tiledb_dim("d2", c(1L, 100L), type = "INT32")))
dim(dom)
```

---

dimensions,tiledb_array_schema-method	
	<i>Returns a list of tiledb_dim objects associated with the tiledb_array_schema</i>

---

**Description**

Returns a list of tiledb\_dim objects associated with the tiledb\_array\_schema

**Usage**

```
## S4 method for signature 'tiledb_array_schema'
dimensions(object)
```

**Arguments**

object	tiledb_array_schema
--------	---------------------

**Value**

a list of tiledb\_dim objects

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32"),
                             tiledb_dim("d2", c(1L, 50L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
dimensions(dom)

lapply(dimensions(dom), name)
```

---

dimensions, tiledb\_domain-method

*Returns a list of the tiledb\_domain dimension objects*

---

**Description**

Returns a list of the tiledb\_domain dimension objects

**Usage**

```
## S4 method for signature 'tiledb_domain'
dimensions(object)
```

**Arguments**

object                    tiledb\_domain

**Value**

a list of tiledb\_dim

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32"),
                             tiledb_dim("d2", c(1L, 50L), type = "INT32")))
dimensions(dom)

lapply(dimensions(dom), name)
```

---

domain*Generic Methods*

---

**Description**

Definition of generic methods

**Usage**

`domain(object, ...)`

`dimensions(object, ...)`

`attrs(object, idx, ...)`

`cell_order(object, ...)`

`tile_order(object, ...)`

`filter_list(object, ...)`

`is.sparse(object, ...)`

`tiledb_ndim(object, ...)`

`name(object)`

`datatype(object)`

`cell_val_num(object)`

`config(object, ...)`

`schema(object, ...)`

`tile(object)`

`is.integral(object)`

`set_max_chunk_size(object, value)`

`max_chunk_size(object)`

`nfilters(object)`

**Arguments**

`object`            A TileDB object

...	Variable argument
idx	An index argument
value	A value to be assigned

---

domain, tiledb\_array\_schema-method

*Returns the tiledb\_domain object associated with a given tiledb\_array\_schema*

---

## Description

Returns the tiledb\_domain object associated with a given tiledb\_array\_schema

## Usage

```
## S4 method for signature 'tiledb_array_schema'
domain(object)
```

## Arguments

object	tiledb_array_schema
--------	---------------------

## Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
domain(sch)
```

---

domain, tiledb\_dim-method

*Return the tiledb\_dim domain*

---

## Description

Return the tiledb\_dim domain

## Usage

```
## S4 method for signature 'tiledb_dim'
domain(object)
```

## Arguments

object	tiledb_dim object
--------	-------------------

**Value**

a vector of (lb, ub) inclusive domain of the dimension

**Examples**

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L))
domain(d1)
```

---

extended

*Retrieve data.frame extended returns columns toggle*

---

**Description**

A tiledb\_array object can be returned as data.frame. This methods returns the selection value for 'extended' format including row (and column, if present) indices.

**Usage**

```
extended(object)

## S4 method for signature 'tiledb_array'
extended(object)
```

**Arguments**

object            A tiledb\_array object

**Value**

A logical value indicating whether an extended return is selected

---

extended<-

*Set data.frame extended return columns toggle*

---

**Description**

A tiledb\_array object can be returned as data.frame. This methods set the selection value for 'extended' format including row (and column, if present) indices.

**Usage**

```
extended(x) <- value

## S4 replacement method for signature 'tiledb_array'
extended(x) <- value
```

**Arguments**

x	A tiledb_array object
value	A logical value with the selection

**Value**

The modified tiledb\_array array object

---

`filter_list, tiledb_array_schema-method`

*Returns the offsets and coordinate filter\_lists associated with the tiledb\_array\_schema*

---

**Description**

Returns the offsets and coordinate filter\_lists associated with the tiledb\_array\_schema

**Usage**

```
## S4 method for signature 'tiledb_array_schema'
filter_list(object)
```

**Arguments**

object	tiledb_array_schema
--------	---------------------

**Value**

a list of tiledb\_filter\_list objects

---

`filter_list, tiledb_attr-method`

*Returns the tiledb\_filter\_list object associated with the given tiledb\_attr*

---

**Description**

Returns the tiledb\_filter\_list object associated with the given tiledb\_attr

**Usage**

```
## S4 method for signature 'tiledb_attr'
filter_list(object)
```



**Arguments**

object            tiledb\_attr

**Value**

a tiledb\_filter\_list object

**Examples**

```
attr <- tiledb_attr(type = "INT32", filter_list=tiledb_filter_list(list(tiledb_filter("ZSTD"))))
filter_list(attr)
```

---

fromDataFrame

*Create a TileDB Dense Array from a given data.frame Object*

---

**Description**

The supplied data.frame object is (currently) limited to integer, numeric, or character columns.

**Usage**

```
fromDataFrame(obj, uri)
```

**Arguments**

obj            A data.frame object.

uri            A character variable with an Array URI.

**Details**

The create (Dense) Array will have as many attributes as there are columns in the data.frame. Each attribute will be a single column.

At present, factor variable are converted to character.

**Value**

Null, invisibly.

## Examples

```
## Not run:
uri <- tempfile()
## turn factor into character
irisdf <- within(iris, Species <- as.character(Species))
fromDataFrame(irisdf, uri)
arr <- tiledb_dense(uri, as.data.frame=TRUE)
newdf <- arr[]
all.equal(iris, newdf)

## End(Not run)
```

---

is.anonymous	<i>Returns TRUE if the tiledb_dim is anonymous</i>
--------------	--

---

## Description

A TileDB attribute is anonymous if no name/label is defined

## Usage

```
is.anonymous(object)

## S3 method for class 'tiledb_attr'
is.anonymous(object)
```

## Arguments

object            tiledb\_attr object

## Value

TRUE or FALSE

## Examples

```
a1 <- tiledb_attr("a1", type = "FLOAT64")
is.anonymous(a1)

a2 <- tiledb_attr("", type = "FLOAT64")
is.anonymous(a2)
```

---

```
is.anonymous.tiledb_dim
```

*Returns TRUE if the tiledb\_dim is anonymous*

---

### Description

A TileDB dimension is anonymous if no name/label is defined

### Usage

```
## S3 method for class 'tiledb_dim'
is.anonymous(object)
```

### Arguments

object                      tiledb\_dim object

### Value

TRUE or FALSE

### Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L), 10L)
is.anonymous(d1)

d2 <- tiledb_dim("", c(1L, 10L), 10L)
is.anonymous(d2)
```

---

```
is.integral,tiledb_domain-method
```

*Returns TRUE is tiledb\_domain is an integral (integer) domain*

---

### Description

Returns TRUE is tiledb\_domain is an integral (integer) domain

### Usage

```
## S4 method for signature 'tiledb_domain'
is.integral(object)
```

### Arguments

object                      tiledb\_domain

**Value**

TRUE if the domain is an integral domain, else FALSE

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32")))
is.integral(dom)
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
is.integral(dom)
```

---

is.sparse,tiledb\_array\_schema-method

*Returns TRUE if the tiledb\_array\_schema is sparse, else FALSE*

---

**Description**

Returns TRUE if the tiledb\_array\_schema is sparse, else FALSE

**Usage**

```
## S4 method for signature 'tiledb_array_schema'
is.sparse(object)
```

**Arguments**

object                    tiledb\_array\_schema

**Value**

TRUE if tiledb\_array\_schema is sparse

---

is.sparse,tiledb\_dense-method

*Returns true is if the array or array\_schema is sparse*

---

**Description**

Returns true is if the array or array\_schema is sparse

**Usage**

```
## S4 method for signature 'tiledb_dense'
is.sparse(object)
```

**Arguments**

object                      tiledb\_dense

**Value**

FALSE

---

is.sparse,tiledb\_sparse-method  
*Check if object is sparse*

---

**Description**

Check if object is sparse

**Usage**

```
## S4 method for signature 'tiledb_sparse'
is.sparse(object)
```

**Arguments**

object                      TileDB object

**Value**

A logical value indicating whether the object is sparse

---

limitTileDBCores            *Limit TileDB core use to a given number of cores*

---

**Description**

By default, TileDB will use all available cores on a given machine. In multi-user or multi-process settings, one may want to reduce the number of core. This function will take a given number, or default to smaller of the ‘Ncpus’ options value or the “OMP\_THREAD\_LIMIT” enviroment variable (or two as hard fallback).

**Usage**

```
limitTileDBCores(ncores, verbose = FALSE)
```

**Arguments**

ncores	Value of CPUs used, if missing the smaller of a fallback of two, the value of ‘Ncpus’ (if set) and the value of environment variable “OMP_THREAD_LIMIT” is used.
verbose	Optional logical toggle; if set, a short message is displayed informing the user about the value set.

**Value**

The modified configuration object is returned invisibly. As a side-effect the updated config is also used to set the global context object.

---

max\_chunk\_size, tiledb\_filter\_list-method

*Returns the filter\_list’s max\_chunk\_size*

---

**Description**

Returns the filter\_list’s max\_chunk\_size

**Usage**

```
## S4 method for signature 'tiledb_filter_list'
max_chunk_size(object)
```

**Arguments**

object	tiledb_filter_list
--------	--------------------

**Value**

integer max\_chunk\_size

**Examples**

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
max_chunk_size(filter_list)
```

---

```
name, tiledb_attr-method
```

*Return the tiledb\_attr name*

---

**Description**

Return the tiledb\_attr name

**Usage**

```
## S4 method for signature 'tiledb_attr'
name(object)
```

**Arguments**

object                      tiledb\_attr object

**Value**

string name, empty string if the attribute is anonymous

**Examples**

```
a1 <- tiledb_attr("a1", type = "INT32")
name(a1)

a2 <- tiledb_attr(type = "INT32")
name(a2)
```

---

```
name, tiledb_dim-method
```

*Return the tiledb\_dim name*

---

**Description**

Return the tiledb\_dim name

**Usage**

```
## S4 method for signature 'tiledb_dim'
name(object)
```

**Arguments**

object                      tiledb\_dim object

**Value**

string name, empty string if the dimension is anonymous

**Examples**

```
d1 <- tiledb_dim("d1", c(1L, 10L))
name(d1)

d2 <- tiledb_dim("", c(1L, 10L))
name(d2)
```

---

nfilters, tiledb\_filter\_list-method

*Returns the filter\_list's number of filters*

---

**Description**

Returns the filter\_list's number of filters

**Usage**

```
## S4 method for signature 'tiledb_filter_list'
nfilters(object)
```

**Arguments**

object            tiledb\_filter\_list

**Value**

integer number of filters

**Examples**

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
nfilters(filter_list)
```



---

```
print.tiledb_metadata Print a TileDB Array Metadata object
```

---

**Description**

Print a TileDB Array Metadata object

**Usage**

```
## S3 method for class 'tiledb_metadata'
print(x, width = NULL, ...)
```

**Arguments**

x	A TileDB array object
width	Optional display width, defaults to NULL
...	Optional method arguments, currently unused

**Value**

The array object, invisibly

---

```
return.data.frame Retrieve data.frame return toggle
```

---

**Description**

A tiledb\_dense object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods returns the selection value.

**Usage**

```
return.data.frame(object, ...)

## S4 method for signature 'tiledb_dense'
return.data.frame(object)
```

**Arguments**

object	A tiledb_dense array object
...	Currently unused

**Value**

A logical value indicating whether data.frame return is selected

---

return.data.frame,tiledb\_array-method  
*Retrieve data.frame return toggle*

---

**Description**

A tiledb\_array object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods returns the selection value.

**Usage**

```
## S4 method for signature 'tiledb_array'
return.data.frame(object)
```

**Arguments**

object            A tiledb\_array object

**Value**

A logical value indicating whether data.frame return is selected

---

return.data.frame,tiledb\_sparse-method  
*Retrieve data.frame return toggle*

---

**Description**

A tiledb\_sparse object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods returns the selection value.

**Usage**

```
## S4 method for signature 'tiledb_sparse'
return.data.frame(object)
```

**Arguments**

object            A tiledb\_sparse array object

**Value**

A logical value indicating whether data.frame return is selected

---

```
return.data.frame<-      Set data.frame return toggle
```

---

**Description**

A tiledb\_dense object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods sets the selection value.

**Usage**

```
return.data.frame(x) <- value

## S4 replacement method for signature 'tiledb_dense'
return.data.frame(x) <- value
```

**Arguments**

x	A tiledb_dense array object
value	A logical value with the selection

**Value**

The modified tiledb\_dense array object

---

```
return.data.frame<-,tiledb_array-method
      Set data.frame return toggle
```

---

**Description**

A tiledb\_array object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods sets the selection value.

**Usage**

```
## S4 replacement method for signature 'tiledb_array'
return.data.frame(x) <- value
```

**Arguments**

x	A tiledb_array object
value	A logical value with the selection

**Value**

The modified tiledb\_array array object

---

```
return.data.frame<-, tiledb_sparse-method
  Set data.frame return toggle
```

---

### Description

A tiledb\_sparse object can be returned as an array (or list of arrays), or, if select, as a data.frame. This methods sets the selection value.

### Usage

```
## S4 replacement method for signature 'tiledb_sparse'
return.data.frame(x) <- value
```

### Arguments

x	A tiledb_sparse array object
value	A logical value with the selection

### Value

The modified tiledb\_sparse array object

---

r_to_tiledb_type	Look up TileDB type corresponding to the type of an R object
------------------	--

---

### Description

Look up TileDB type corresponding to the type of an R object

### Usage

```
r_to_tiledb_type(x)
```

### Arguments

x	an R array or list
---	--------------------

### Value

single character, e.g. INT32

---

schema, tiledb\_array-method

*Return a schema from a tiledb\_array object*

---

### Description

Return a schema from a tiledb\_array object

### Usage

```
## S4 method for signature 'tiledb_array'
schema(object, ...)
```

### Arguments

object	tiledb array object
...	Extra parameter for function signature, currently unused

### Value

The scheme for the object

---

schema, tiledb\_dense-method

*Returns the tiledb\_dense array tiledb\_schema object*

---

### Description

Returns the tiledb\_dense array tiledb\_schema object

### Usage

```
## S4 method for signature 'tiledb_dense'
schema(object, ...)
```

### Arguments

object	tiledb_dense array object
...	Extra parameter for method signature, currently unused.

### Value

tiledb\_schema

---

schema, tiledb\_sparse-method

*Return a schema from a sparse array*

---

### Description

Return a schema from a sparse array

### Usage

```
## S4 method for signature 'tiledb_sparse'
schema(object, ...)
```

### Arguments

object	sparse array object
...	Extra parameter for function signature, currently unused

### Value

The scheme for the object

---

selected\_ranges

*Retrieve selected\_ranges values for the array*

---

### Description

A tiledb\_array object can have a range selection for each dimension attribute. This methods returns the selection value for 'selected\_ranges' and returns a list (with one element per dimension) of two-column matrices where each row describes one pair of minimum and maximum values.

### Usage

```
selected_ranges(object)

## S4 method for signature 'tiledb_array'
selected_ranges(object)
```

### Arguments

object	A tiledb_array object
--------	-----------------------

### Value

A list which can contain a matrix for each dimension

---

```
selected_ranges<-      Set selected_ranges return values for the array
```

---

### Description

A tiledb\_array object can have a range selection for each dimension attribute. This methods sets the selection value for 'selected\_ranges' which is a list (with one element per dimension) of two-column matrices where each row describes one pair of minimum and maximum values.

### Usage

```
selected_ranges(x) <- value

## S4 replacement method for signature 'tiledb_array'
selected_ranges(x) <- value
```

### Arguments

x	A tiledb_array object
value	A list of two-column matrices where each list element 'i' corresponds to the dimension attribute 'i'. The matrices can contain rows where each row contains the minimum and maximum value of a range.

### Value

The modified tiledb\_array array object

---

```
set_max_chunk_size,tiledb_filter_list,numeric-method
      Set the filter_list's max_chunk_size
```

---

### Description

Set the filter\_list's max\_chunk\_size

### Usage

```
## S4 method for signature 'tiledb_filter_list,numeric'
set_max_chunk_size(object, value)
```

### Arguments

object	tiledb_filter_list
value	string

## Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
set_max_chunk_size(filter_list, 10)
```

---

show, tiledb\_array-method

*Prints a tiledb\_array object*

---

## Description

Prints a tiledb\_array object

## Usage

```
## S4 method for signature 'tiledb_array'
show(object)
```

## Arguments

object                    A tiledb array object

---

show, tiledb\_array\_schema-method

*Prints an array schema object*

---

## Description

Prints an array schema object

## Usage

```
## S4 method for signature 'tiledb_array_schema'
show(object)
```

## Arguments

object                    An array\_schema object



---

```
show, tiledb_attr-method
```

*Prints an attribute object*

---

## Description

Prints an attribute object

## Usage

```
## S4 method for signature 'tiledb_attr'  
show(object)
```

## Arguments

object	An attribute object
--------	---------------------

---

```
show, tiledb_config-method
```

*Prints the config object to STDOUT*

---

## Description

Prints the config object to STDOUT

## Usage

```
## S4 method for signature 'tiledb_config'  
show(object)
```

## Arguments

object	tiledb_config object
--------	----------------------

## Examples

```
cfg <- tiledb_config()  
show(cfg)
```

---

show,tiledb\_dense-method

*Prints a tiledb\_dense array object*

---

### Description

Prints a tiledb\_dense array object

### Usage

```
## S4 method for signature 'tiledb_dense'  
show(object)
```

### Arguments

object	A tiledb_dense array object
--------	-----------------------------

---

show,tiledb\_domain-method

*Prints an domain object*

---

### Description

Prints an domain object

### Usage

```
## S4 method for signature 'tiledb_domain'  
show(object)
```

### Arguments

object	An domain object
--------	------------------

---

`show, tiledb_sparse-method`*Prints a tiledb\_sparse array object*

---

**Description**

Prints a tiledb\_sparse array object

**Usage**

```
## S4 method for signature 'tiledb_sparse'  
show(object)
```

**Arguments**

object            A tiledb\_sparse array object

---

`tile, tiledb_dim-method`*Return the tiledb\_dim tile extent*

---

**Description**

Return the tiledb\_dim tile extent

**Usage**

```
## S4 method for signature 'tiledb_dim'  
tile(object)
```

**Arguments**

object            tiledb\_dim object

**Value**

a scalar tile extent

**Examples**

```
d1 <- tiledb_dim("d1", domain = c(5L, 10L), tile = 2L)  
tile(d1)
```

---

tiledb_array	<i>Constructs a tiledb_array object backed by a persisted tiledb array uri</i>
--------------	--

---

## Description

tiledb\_array returns a new object. This class is experimental.

## Usage

```
tiledb_array(
  uri,
  query_type = c("READ", "WRITE"),
  is.sparse = NA,
  as.data.frame = FALSE,
  attrs = character(),
  extended = TRUE,
  selected_ranges = list(),
  ctx = tiledb_get_context()
)
```

## Arguments

uri	uri path to the tiledb dense array
query_type	optionally loads the array in "READ" or "WRITE" only modes.
is.sparse	optional logical switch, defaults to "NA" letting array determine it
as.data.frame	optional logical switch, defaults to "FALSE"
attrs	optional character vector to select attributes, default is empty implying all are selected
extended	optional logical switch selecting wide 'data.frame' format, defaults to "TRUE"
selected_ranges	An optional list with matrices where each matrix i describes the (min,max) pair of ranges for dimension i
ctx	tiledb_ctx (optional)

## Value

tiledb\_sparse array object

---

tiledb_array-class	<i>An S4 class for a TileDB Array</i>
--------------------	---------------------------------------

---

### Description

This class aims to eventually replace [tiledb\\_dense](#) and [tiledb\\_sparse](#) provided equivalent functionality based on refactored implementation utilising newer TileDB features.

### Slots

ctx A TileDB context object  
uri A character desription  
is.sparse A logical value  
as.data.frame A logical value  
attrs A character vector  
extended A logical value  
selected\_ranges An optional list with matrices where each matrix i describes the (min,max) pair of ranges for dimension i  
ptr External pointer to the underlying implementation

---

tiledb_array_close	<i>Close a TileDB Array</i>
--------------------	-----------------------------

---

### Description

Close a TileDB Array

### Usage

```
tiledb_array_close(arr)
```

### Arguments

arr A TileDB Array object as for example returned by [tiledb\\_dense\(\)](#)

### Value

The TileDB Array object but closed

---

tiledb_array_create	<i>Creates a new TileDB array given an input schema.</i>
---------------------	--

---

**Description**

Creates a new TileDB array given an input schema.

**Usage**

```
tiledb_array_create(uri, schema)
```

**Arguments**

uri	URI specifying path to create the TileDB array object
schema	tiledb_array_schema object

**Examples**

```
## Not run:
pth <- tempdir()
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32")))
tiledb_array_create(pth, sch)
tiledb_object_type(pth)

## End(Not run)
```

---

tiledb_array_is_heterogeneous	<i>Check for Heterogeneous Domain</i>
-------------------------------	---------------------------------------

---

**Description**

Check for Heterogeneous Domain

**Usage**

```
tiledb_array_is_heterogeneous(arr)
```

**Arguments**

arr	A TileDB Array object
-----	-----------------------

**Value**

A boolean indicating if the array has heterogenous domains

---

tiledb\_array\_is\_homogeneous  
*Check for Homogeneous Domain*

---

### Description

Check for Homogeneous Domain

### Usage

tiledb\_array\_is\_homogeneous(arr)

### Arguments

arr                    A TileDB Array object

### Value

A boolean indicating if the array has homogeneous domains

---

tiledb\_array\_open        *Open a TileDB Array*

---

### Description

Open a TileDB Array

### Usage

tiledb\_array\_open(arr, type = c("READ", "WRITE"))

### Arguments

arr                    A TileDB Array object as for example returned by tiledb\_dense()  
 type                  A character value that must be either 'READ' or 'WRITE'

### Value

The TileDB Array object but opened for reading or writing

---

tiledb_array_schema	<i>Constructs a tiledb_array_schema object</i>
---------------------	--

---

## Description

Constructs a tiledb\_array\_schema object

## Usage

```
tiledb_array_schema(
  domain,
  attrs,
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  sparse = FALSE,
  coords_filter_list = NULL,
  offsets_filter_list = NULL,
  ctx = tiledb_get_context()
)
```

## Arguments

domain	tiledb_domain object
attrs	a list of one or more tiledb_attr objects
cell_order	(default "COL_MAJOR")
tile_order	(default "COL_MAJOR")
sparse	(default FALSE)
coords_filter_list	(optional)
offsets_filter_list	(optional)
ctx	tiledb_ctx object (optional)

## Examples

```
schema <- tiledb_array_schema(
  dom = tiledb_domain(
    dims = c(tiledb_dim("rows", c(1L, 4L), 4L, "INT32"),
             tiledb_dim("cols", c(1L, 4L), 4L, "INT32"))),
  attrs = c(tiledb_attr("a", type = "INT32")),
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  sparse = FALSE)

schema
```



---

tiledb_array_schema-class	<i>An S4 class for the TileDB array schema</i>
---------------------------	--

---

**Description**

An S4 class for the TileDB array schema

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb_attr	<i>Constructs a tiledb_attr object</i>
-------------	--

---

**Description**

Constructs a tiledb\_attr object

**Usage**

```
tiledb_attr(  
  name,  
  type,  
  filter_list = tiledb_filter_list(),  
  ncells = 1,  
  ctx = tiledb_get_context()  
)
```

**Arguments**

name	The dimension name / label string; if missing default "" is used.
type	The tiledb_attr TileDB datatype string; if missing the user is alerted that this is a <i>required</i> parameter.
filter_list	(default filter_list("NONE")) The tiledb_attr filter_list
ncells	(default 1) The number of cells, use NA to signal variable length
ctx	tiledb_ctx object (optional)

**Value**

tiledb\_dim object

**Examples**

```
flt <- tiledb_filter_list(list(tiledb_filter("GZIP")))
attr <- tiledb_attr(name = "a1", type = "INT32",
                    filter_list = flt)
attr
```

---

tiledb_attr-class	<i>An S4 class for a TileDB attribute</i>
-------------------	---

---

**Description**

An S4 class for a TileDB attribute

**Slots**

ptr External pointer to the underlying implementation

---

tiledb_config	<i>Creates a tiledb_config object</i>
---------------	---------------------------------------

---

**Description**

Creates a tiledb\_config object

**Usage**

```
tiledb_config(config = NA_character_)
```

**Arguments**

config (optional) character vector of config parameter names, values

**Value**

tiledb\_config object

**Examples**

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]

# set tile cache size to custom value
cfg <- tiledb_config(c("sm.tile_cache_size" = "100"))
cfg["sm.tile_cache_size"]
```

---

tiledb_config-class	<i>An S4 class for a TileDB configuration</i>
---------------------	---

---

### Description

An S4 class for a TileDB configuration

### Slots

ptr An external pointer to the underlying implementation

---

tiledb_config_load	<i>Load a saved tiledb_config file from disk</i>
--------------------	--

---

### Description

Load a saved tiledb\_config file from disk

### Usage

```
tiledb_config_load(path)
```

### Arguments

path	path to the config file
------	-------------------------

### Examples

```
tmp <- tempfile()
cfg <- tiledb_config(c("sm.tile_cache_size" = "10"))
pth <- tiledb_config_save(cfg, tmp)
cfg <- tiledb_config_load(pth)
cfg["sm.tile_cache_size"]
```

---

tiledb_config_save	<i>Save a tiledb_config object ot a local text file</i>
--------------------	---

---

**Description**

Save a tiledb\_config object ot a local text file

**Usage**

```
tiledb_config_save(config, path)
```

**Arguments**

config	The tiledb_config object
path	The path to config file to be created

**Value**

path to created config file

**Examples**

```
tmp <- tempfile()
cfg <- tiledb_config(c("sm.tile_cache_size" = "10"))
pth <- tiledb_config_save(cfg, tmp)

cat(readLines(pth), sep = "\n")
```

---

tiledb_ctx	<i>Creates a tiledb_ctx object</i>
------------	------------------------------------

---

**Description**

Creates a tiledb\_ctx object

**Usage**

```
tiledb_ctx(config = NULL, cached = TRUE)
```

**Arguments**

config	(optional) character vector of config parameter names, values
cached	(optional) logical switch to force new creation

**Value**

tiledb\_ctx object

**Examples**

```
# default configuration
ctx <- tiledb_ctx()

# optionally set config parameters
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "100"))
```

---

tiledb_ctx-class	<i>An S4 class for a TileDB context</i>
------------------	---

---

**Description**

An S4 class for a TileDB context

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb_ctx_set_default_tags	<i>Sets default context tags</i>
-----------------------------	----------------------------------

---

**Description**

Sets default context tags

**Usage**

```
tiledb_ctx_set_default_tags(object)
```

**Arguments**

object tiledb\_ctx object

---

tiledb_ctx_set_tag	<i>Sets a string:string "tag" on the Ctx</i>
--------------------	--

---

### Description

Sets a string:string "tag" on the Ctx

### Usage

```
tiledb_ctx_set_tag(object, key, value)
```

### Arguments

object	tiledb_ctx object
key	string
value	string

### Examples

```
ctx <- tiledb_ctx(c("sm.tile_cache_size" = "10"))
cfg <- tiledb_ctx_set_tag(ctx, "tag", "value")
```

---

tiledb_delete_metadata	<i>Delete a TileDB Array Metadata object given by key</i>
------------------------	---

---

### Description

Delete a TileDB Array Metadata object given by key

### Usage

```
tiledb_delete_metadata(arr, key)
```

### Arguments

arr	A TileDB Array object
key	A character value describing a metadata key

### Value

A boolean indicating success

---

tiledb_dense	<i>Constructs a tiledb_dense object backed by a persisted tiledb array uri</i>
--------------	--

---

**Description**

Constructs a tiledb\_dense object backed by a persisted tiledb array uri

**Usage**

```
tiledb_dense(  
  uri,  
  query_type = c("READ", "WRITE"),  
  as.data.frame = FALSE,  
  attrs = character(),  
  extended = FALSE,  
  ctx = tiledb_get_context()  
)
```

**Arguments**

uri	uri path to the tiledb dense array
query_type	optionally loads the array in "READ" or "WRITE" only modes.
as.data.frame	optional logical switch, defaults to "FALSE"
attrs	optional character vector to select attributes, default is empty implying all are selected
extended	optional logical switch selecting wide 'data.frame' format, defaults to "FALSE"
ctx	tiledb_ctx (optional)

**Value**

tiledb\_dense array object

---

tiledb_dense-class	<i>An S4 class for a TileDB dense array</i>
--------------------	---

---

**Description**

An S4 class for a TileDB dense array

**Slots**

ctx A TileDB context object  
 uri A character description  
 as.data.frame A logical value  
 attrs A character vector  
 extended A logical value  
 ptr External pointer to the underlying implementation

---

tiledb_dim	<i>Constructs a tiledb_dim object</i>
------------	---------------------------------------

---

**Description**

Constructs a tiledb\_dim object

**Usage**

```
tiledb_dim(name, domain, tile, type, ctx = tiledb_get_context())
```

**Arguments**

name	The dimension name / label string. This argument is required.
domain	The dimension (inclusive) domain. The dimension's domain is defined by a (lower bound, upper bound) vector, and is usually either of type integer or double (i.e. numeric). For type, ASCII NULL is expected.
tile	The tile dimension tile extent. For type, ASCII NULL is expected.
type	The dimension TileDB datatype string
ctx	tiledb_ctx object (optional)

**Value**

tiledb\_dim object

**Examples**

```
tiledb_dim(name = "d1", domain = c(1L, 10L), tile = 5L, type = "INT32")
```



---

tiledb_dim-class	<i>An S4 class for a TileDB dimension object</i>
------------------	--

---

### Description

An S4 class for a TileDB dimension object

### Slots

ptr An external pointer to the underlying implementation

---

tiledb_domain	<i>Constructs a tiledb_domain object</i>
---------------	--

---

### Description

All tiledb\_dim must be of the same TileDB type.

### Usage

```
tiledb_domain(dims, ctx = tiledb_get_context())
```

### Arguments

dims	list() of tiledb_dim objects
ctx	tiledb_ctx (optional)

### Value

tiledb\_domain

### Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 100L), type = "INT32"),
                             tiledb_dim("d2", c(1L, 50L), type = "INT32")))
```

---

tiledb_domain-class	<i>An S4 class for a TileDB domain</i>
---------------------	--

---

### Description

An S4 class for a TileDB domain

### Slots

ptr External pointer to the underlying implementation

---

tiledb_filter	<i>Constructs a tiledb_filter object</i>
---------------	--

---

**Description**

Available filters:

- "NONE"
- "GZIP"
- "ZSTD"
- "LZ4"
- "RLE"
- "BZIP2"
- "DOUBLE\_DELTA"
- "BIT\_WIDTH\_REDUCTION"
- "BITSHUFFLE"
- "BYTESHUFFLE"
- "POSITIVE\_DELTA"

**Usage**

```
tiledb_filter(name = "NONE", ctx = tiledb_get_context())
```

**Arguments**

name	(default "NONE") TileDB filter name string
ctx	tiledb_ctx object (optional)

**Details**

Valid compression options vary depending on the filter used, consult the TileDB docs for more information.

**Value**

tiledb\_filter object

**Examples**

```
tiledb_filter("ZSTD")
```

---

tiledb\_filter-class     *An S4 class for a TileDB filter*

---

## Description

An S4 class for a TileDB filter

## Slots

ptr External pointer to the underlying implementation

---

tiledb\_filter\_get\_option  
*Returns the filter's option*

---

## Description

Returns the filter's option

## Usage

```
tiledb_filter_get_option(object, option)
```

## Arguments

object	tiledb_filter
option	string

## Value

Integer value

## Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_set_option(c,"COMPRESSION_LEVEL", 5)
tiledb_filter_get_option(c, "COMPRESSION_LEVEL")
```

---

tiledb_filter_list	<i>Constructs a tiledb_filter_list object</i>
--------------------	---

---

**Description**

Constructs a tiledb\_filter\_list object

**Usage**

```
tiledb_filter_list(filters = c(), ctx = tiledb_get_context())
```

**Arguments**

filters	an optional list of one or more tiledb_filter_list objects
ctx	tiledb_ctx object (optional)

**Value**

tiledb\_filter\_list object

**Examples**

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
filter_list
```

---

tiledb_filter_list-class	<i>An S4 class for a TileDB filter list</i>
--------------------------	---

---

**Description**

An S4 class for a TileDB filter list

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb\_filter\_set\_option  
*Set the filter's option*

---

### Description

Set the filter's option

### Usage

```
tiledb_filter_set_option(object, option, value)
```

### Arguments

object	tiledb_filter
option	string
value	int

### Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_set_option(c, "COMPRESSION_LEVEL", 5)
tiledb_filter_get_option(c, "COMPRESSION_LEVEL")
```

---

tiledb\_filter\_type      *Returns the type of the filter used*

---

### Description

Returns the type of the filter used

### Usage

```
tiledb_filter_type(object)
```

### Arguments

object	tiledb_filter
--------	---------------

### Value

TileDB filter type string

## Examples

```
c <- tiledb_filter("ZSTD")
tiledb_filter_type(c)
```

---

tiledb\_get\_all\_metadata

*Return a TileDB Array Metadata object given by key*

---

## Description

Return a TileDB Array Metadata object given by key

## Usage

```
tiledb_get_all_metadata(arr)
```

## Arguments

arr                    A TileDB Array object, or a character URI describing one

## Value

A object stored in the Metadata under the given key

---

tiledb\_get\_context

*Retrieve a TileDB context object from the package cache*

---

## Description

Retrieve a TileDB context object from the package cache

## Usage

```
tiledb_get_context()
```

## Value

A TileDB context object

---

tiledb_get_metadata	<i>Return a TileDB Array Metadata object given by key</i>
---------------------	---

---

### Description

Return a TileDB Array Metadata object given by key

### Usage

```
tiledb_get_metadata(arr, key)
```

### Arguments

arr	A TileDB Array object, or a character URI describing one
key	A character value describing a metadata key

### Value

A object stored in the Metadata under the given key, or 'NULL' if none found.

---

tiledb_group_create	<i>Creates a TileDB group object at given uri path</i>
---------------------	--

---

### Description

Creates a TileDB group object at given uri path

### Usage

```
tiledb_group_create(uri, ctx = tiledb_get_context())
```

### Arguments

uri	path which to create group
ctx	tiledb_ctx object (optional)

### Value

uri of created group

**Examples**

```
## Not run:
pth <- tempdir()
tiledb_group_create(pth)
tiledb_object_type(pth)

## End(Not run)
```

---

tiledb_has_metadata	<i>Test if TileDB Array has Metadata</i>
---------------------	--

---

**Description**

Test if TileDB Array has Metadata

**Usage**

```
tiledb_has_metadata(arr, key)
```

**Arguments**

- arr                   A TileDB Array object
- key                   A character value describing a metadata key

**Value**

A logical value indicating if the given key exists in the metdata of the given array

---

tiledb_is_supported_fs	<i>Query if a TileDB backend is supported</i>
------------------------	---

---

**Description**

The scheme corresponds to the URI scheme for TileDB resouces.

**Usage**

```
tiledb_is_supported_fs(scheme, object = tiledb_get_context())
```

**Arguments**

- scheme               URI string scheme ("file", "hdfs", "s3")
- object               tiledb\_ctx object



## Details

Ex:

- {file}:///path/to/file
- {hdfs}:///path/to/file
- {s3}://hostname:port/path/to/file

## Value

TRUE if tiledb backend is supported, FALSE otherwise

## Examples

```
tiledb_is_supported_fs("file")
tiledb_is_supported_fs("s3")
```

---

tiledb\_ndim,tiledb\_array\_schema-method

*Return the number of dimensions associated with the tiledb\_array\_schema*

---

## Description

Return the number of dimensions associated with the tiledb\_array\_schema

## Usage

```
## S4 method for signature 'tiledb_array_schema'
tiledb_ndim(object)
```

## Arguments

object                    tiledb\_array\_schema

## Value

integer number of dimensions

## Examples

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(1L, 10L), type = "INT32")))
sch <- tiledb_array_schema(dom, attrs = c(tiledb_attr("a1", type = "INT32"),
                                           tiledb_attr("a2", type = "FLOAT64")))
tiledb_ndim(sch)
```

---

tiledb\_ndim,tiledb\_dim-method

*Returns the number of dimensions for a tiledb domain object*


---

### Description

Returns the number of dimensions for a tiledb domain object

### Usage

```
## S4 method for signature 'tiledb_dim'
tiledb_ndim(object)
```

### Arguments

object                    tiledb\_ndim object

### Value

1L

### Examples

```
d1 <- tiledb_dim("d1", c(1L, 10L), 10L)
tiledb_ndim(d1)
```

---

tiledb\_ndim,tiledb\_domain-method

*Returns the number of dimensions of the tiledb\_domain*


---

### Description

Returns the number of dimensions of the tiledb\_domain

### Usage

```
## S4 method for signature 'tiledb_domain'
tiledb_ndim(object)
```

### Arguments

object                    tiledb\_domain

**Value**

integer number of dimensions

**Examples**

```
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64")))
tiledb_ndim(dom)
dom <- tiledb_domain(dims = c(tiledb_dim("d1", c(0.5, 100.0), type = "FLOAT64"),
                             tiledb_dim("d2", c(0.5, 100.0), type = "FLOAT64")))
tiledb_ndim(dom)
```

---

tiledb_num_metadata	<i>Return count of TileDB Array Metadata objects</i>
---------------------	--

---

**Description**

Return count of TileDB Array Metadata objects

**Usage**

```
tiledb_num_metadata(arr)
```

**Arguments**

arr                    A TileDB Array object, or a character URI describing one

**Value**

A integer variable with the number of Metadata objects

---

tiledb_object_ls	<i>List TileDB resources at a given root URI path</i>
------------------	---

---

**Description**

List TileDB resources at a given root URI path

**Usage**

```
tiledb_object_ls(uri, filter = NULL, ctx = tiledb_get_context())
```

**Arguments**

- uri                   uri path to walk
- filter               optional filtering argument, default is "NULL", currently unused
- ctx                   tiledb\_ctx object (optional)

**Value**

a dataframe with object type, object uri string columns

---

tiledb_object_mv	<i>Move a TileDB resource to new uri path</i>
------------------	---

---

**Description**

Raises an error if either uri is invalid, or the old uri resource is not a tiledb object

**Usage**

```
tiledb_object_mv(old_uri, new_uri, ctx = tiledb_get_context())
```

**Arguments**

- old\_uri               old uri of existing tiledb resource
- new\_uri               new uri to move tiledb resource
- ctx                   tiledb\_ctx object (optional)

**Value**

new uri of moved tiledb resource

---

tiledb_object_rm	<i>Removes a TileDB resource</i>
------------------	----------------------------------

---

**Description**

Raises an error if the uri is invalid, or the uri resource is not a tiledb object

**Usage**

```
tiledb_object_rm(uri, ctx = tiledb_get_context())
```

**Arguments**

- uri                   path which to create group
- ctx                   tiledb\_ctx object (optional)

**Value**

uri of removed TileDB resource

---

tiledb_object_type	<i>Return the TileDB object type string of a TileDB resource</i>
--------------------	--

---

**Description**

Object types:

- "ARRAY", dense or sparse TileDB array
- "GROUP", TileDB group
- "INVALID", not a TileDB resource

**Usage**

```
tiledb_object_type(uri, ctx = tiledb_get_context())
```

**Arguments**

uri	path to TileDB resource
ctx	tiledb_ctx object (optional)

**Value**

TileDB object type string

---

tiledb_object_walk	<i>Recursively discover TileDB resources at a given root URI path</i>
--------------------	---

---

**Description**

Recursively discover TileDB resources at a given root URI path

**Usage**

```
tiledb_object_walk(uri, order = "PREORDER", ctx = tiledb_get_context())
```

**Arguments**

uri	root uri path to walk
order	(default "PREORDER") specify "POSTORDER" for "POSTORDER" traversal
ctx	tiledb_ctx object (optional)

**Value**

a dataframe with object type, object uri string columns

---

tiledb_put_metadata	<i>Store an object in TileDB Array Metadata under given key</i>
---------------------	---

---

**Description**

Store an object in TileDB Array Metadata under given key

**Usage**

```
tiledb_put_metadata(arr, key, val)
```

**Arguments**

arr	A TileDB Array object, or a character URI describing one
key	A character value describing a metadata key
val	An object to be store

**Value**

A boolean value indicating success

---

tiledb_query	<i>Creates a 'tiledb_query' object</i>
--------------	--

---

**Description**

Creates a 'tiledb\_query' object

**Usage**

```
tiledb_query(array, type = c("READ", "WRITE"), ctx = tiledb_get_context())
```

**Arguments**

array	A TileDB Array object
type	A character value that must be one of 'READ' or 'WRITE'
ctx	(optional) A TileDB Ctx object

**Value**

'tiledb\_query' object

---

tiledb_query-class	<i>An S4 class for a TileDB Query object</i>
--------------------	--

---

### Description

An S4 class for a TileDB Query object

### Slots

ptr An external pointer to the underlying implementation

---

tiledb_query_add_range	<i>Set a range for a given query</i>
------------------------	--------------------------------------

---

### Description

Set a range for a given query

### Usage

```
tiledb_query_add_range(query, schema, attr, lowval, highval, stride = NULL)
```

### Arguments

query	A TileDB Query object
schema	A TileDB Schema object
attr	An character variable with a dimension name for which the range is set
lowval	The lower value of the range to be set
highval	The higher value of the range to be set
stride	An optional stride value for the range to be set

### Value

The query object, invisibly

---

tiledb\_query\_add\_range\_with\_type  
*Set a range for a given query*

---

### Description

Set a range for a given query

### Usage

```
tiledb_query_add_range_with_type(
    query,
    idx,
    datatype,
    lowval,
    highval,
    stride = NULL
)
```

### Arguments

query	A TileDB Query object
idx	An integer index, zero based, of the dimensions
datatype	A character value containing the data type
lowval	The lower value of the range to be set
highval	The highre value of the range to be set
stride	An optional stride value for the range to be set

### Value

The query object, invisibly

---

tiledb\_query\_alloc\_buffer\_ptr\_char  
*Allocate a Query buffer for reading a character attribute*

---

### Description

Allocate a Query buffer for reading a character attribute

### Usage

```
tiledb_query_alloc_buffer_ptr_char(sizeoffsets, sizedata)
```



### Arguments

sizeoffsets	An optional value of the size of the offsets vector
sizedata	An optional value of the size of the data string

### Value

An external pointer to the allocated buffer object

---

tiledb\_query\_alloc\_buffer\_ptr\_char\_subarray

*Allocate a Query buffer for reading a character attribute using a subarray*

---

### Description

Note that this uses an API part that may be deprecated in the future.

### Usage

```
tiledb_query_alloc_buffer_ptr_char_subarray(
    array,
    attr,
    subarray = NULL,
    sizeoffsets = 0,
    sizedata = 0
)
```

### Arguments

array	A TileDB Array object
attr	A character value containing the attribute
subarray	A vector of length four describing the subarray required for dense arrays
sizeoffsets	An optional value of the size of the offsets vector
sizedata	An optional value of the size of the data string

### Value

An external pointer to the allocated buffer object

---

tiledb\_query\_buffer\_alloc\_ptr  
*Allocate a Query buffer for a given type*

---

### Description

This function allocates a query buffer for the given data type.

### Usage

```
tiledb_query_buffer_alloc_ptr(query, datatype, ncells)
```

### Arguments

query	A TileDB Query object
datatype	A character value containing the data type
ncells	A number of elements (not bytes)

### Value

An external pointer to the allocated buffer object

---

tiledb\_query\_create\_buffer\_ptr  
*Allocate and populate a Query buffer for a given object of a given data type.*

---

### Description

This function allocates a query buffer for the given data object of the given type and assigns the object content to the buffer.

### Usage

```
tiledb_query_create_buffer_ptr(query, datatype, object)
```

### Arguments

query	A TileDB Query object
datatype	A character value containing the data type
object	A vector object of the given type

### Value

An external pointer to the allocated buffer object

---

tiledb_query_create_buffer_ptr_char
<i>Allocate and populate a Query buffer for writing the given char vector</i>

---

**Description**

Allocate and populate a Query buffer for writing the given char vector

**Usage**

tiledb\_query\_create\_buffer\_ptr\_char(query, varvec)

**Arguments**

- |        |                       |
|--------|-----------------------|
| query  | A TileDB Query object |
| varvec | A vector of strings   |

**Value**

An external pointer to the allocated buffer object

---

tiledb_query_finalize	<i>Finalize TileDB Query</i>
-----------------------	------------------------------

---

**Description**

Finalize TileDB Query

**Usage**

tiledb\_query\_finalize(query)

**Arguments**

- |       |                       |
|-------|-----------------------|
| query | A TileDB Query object |
|-------|-----------------------|

**Value**

A character value, either 'READ' or 'WRITE'

---

tiledb\_query\_get\_buffer\_char

*Retrieve content from a Query character buffer*


---

### Description

This function uses a query buffer for a character attribute or dimension and returns its content.

### Usage

```
tiledb_query_get_buffer_char(bufptr, sizeoffsets = 0, sizestring = 0)
```

### Arguments

bufptr	An external pointer with a query buffer
sizeoffsets	An optional argument for the length of the internal offsets vector
sizestring	An optional argument for the length of the internal string

### Value

An R object as resulting from the query

---

tiledb\_query\_get\_buffer\_ptr

*Retrieve content from a Query buffer*


---

### Description

This function uses a query buffer and returns its content.

### Usage

```
tiledb_query_get_buffer_ptr(bufptr)
```

### Arguments

bufptr	An external pointer with a query buffer
--------	---

### Value

An R object as resulting from the query

---

tiledb_query_get_layout
<i>Get TileDB Query layout</i>

---

**Description**

Get TileDB Query layout

**Usage**

tiledb\_query\_get\_layout(query)

**Arguments**

query                    A TileDB Query object

**Value**

The TileDB Query layout as a string

---

tiledb_query_result_buffer_elements
<i>Get TileDB Query result buffer elements</i>

---

**Description**

Get TileDB Query result buffer elements

**Usage**

tiledb\_query\_result\_buffer\_elements(query, attr)

**Arguments**

query                    A TileDB Query object  
attr                    A character value containing the attribute

**Value**

A integer with the number of elements in the results buffer for the given attribute

---

tiledb\_query\_set\_buffer

*Set TileDB Query buffer*


---

### Description

This function allocates query buffers directly from R vectors in case the types match: integer, double, logical. For more general types see tiledb\_query\_buffer\_alloc\_ptr.

### Usage

```
tiledb_query_set_buffer(query, attr, buffer)
```

### Arguments

query	A TileDB Query object
attr	A character value containing the attribute
buffer	A vector providing the query buffer

### Value

The modified query object, invisibly

---

tiledb\_query\_set\_buffer\_ptr

*Assigns to a Query buffer for a given attribute*


---

### Description

This function assigns a given query buffer to a query.

### Usage

```
tiledb_query_set_buffer_ptr(query, attr, bufptr)
```

### Arguments

query	A TileDB Query object
attr	A character value containing the attribute
bufptr	An external pointer with a query buffer

### Value

The modified query object, invisibly

---

tiledb_query_set_buffer_ptr_char
<i>Assign a buffer to a Query attribute</i>

---

**Description**

Assign a buffer to a Query attribute

**Usage**

```
tiledb_query_set_buffer_ptr_char(query, attr, bufptr)
```

**Arguments**

- |        |  |
|--------|--|
| query  | A TileDB Query object                      |
| attr   | A character value containing the attribute |
| bufptr | An external pointer with a query buffer    |

**Value**

The modified query object, invisibly

---

tiledb_query_set_layout
<i>Set TileDB Query layout</i>

---

**Description**

Set TileDB Query layout

**Usage**

```
tiledb_query_set_layout(  
  query,  
  layout = c("ROW_MAJOR", "COL_MAJOR", "GLOBAL_ORDER", "UNORDERED")  
)
```

**Arguments**

- |        |   |
|--------|---|
| query  | A TileDB Query object   |
| layout | A character variable with the layout; must be one of "ROW_MAJOR", "COL_MAJOR", "GLOBAL_ORDER", "UNORDERED") |

**Value**

The modified query object, invisibly

---

tiledb_query_set_subarray	<i>Set subarray for TileDB Query object</i>
---------------------------	---

---

**Description**

Set subarray for TileDB Query object

**Usage**

tiledb\_query\_set\_subarray(query, subarray, type)

**Arguments**

- |          |   |
|----------|---|
| query    | A TileDB Query object   |
| subarray | A subarray vector object  |
| type     | An optional type as a character, if missing type is inferred from the vector. |

**Value**

The modified query object, invisibly

---

tiledb_query_status	<i>Get TileDB Query status</i>
---------------------	--------------------------------

---

**Description**

Get TileDB Query status

**Usage**

tiledb\_query\_status(query)

**Arguments**

- |       |                       |
|-------|-----------------------|
| query | A TileDB Query object |
|-------|-----------------------|

**Value**

A character value describing the query status



---

tiledb_query_submit	<i>Submit TileDB Query</i>
---------------------	----------------------------

---

**Description**

Submit TileDB Query

**Usage**

tiledb\_query\_submit(query)

**Arguments**

query                    A TileDB Query object

**Value**

The modified query object, invisibly

---

tiledb_query_type	<i>Return TileDB Query type</i>
-------------------	---------------------------------

---

**Description**

Return TileDB Query type

**Usage**

tiledb\_query\_type(query)

**Arguments**

query                    A TileDB Query object

**Value**

A character value, either 'READ' or 'WRITE'

---

`tiledb_schema_get_names`*Get all Dimension and Attribute Names*

---

**Description**

Get all Dimension and Attribute Names

**Usage**

```
tiledb_schema_get_names(sch)
```

**Arguments**

`sch`                      A TileDB Schema object

**Value**

A character vector of dimension and attribute names

---

`tiledb_schema_get_types`*Get all Dimension and Attribute Types*

---

**Description**

Get all Dimension and Attribute Types

**Usage**

```
tiledb_schema_get_types(sch)
```

**Arguments**

`sch`                      A TileDB Schema object

**Value**

A character vector of dimension and attribute data types

---

tiledb_set_context	<i>Store a TileDB context object in the package cache</i>
--------------------	---

---

### Description

Store a TileDB context object in the package cache

### Usage

```
tiledb_set_context(ctx)
```

### Arguments

ctx	A TileDB context object
-----	-------------------------

### Value

A TileDB context object

---

tiledb_sparse	<i>Constructs a tiledb_sparse object backed by a persisted tiledb array uri</i>
---------------	---

---

### Description

tiledb\_sparse returns a list of coordinates and attributes vectors for reads

### Usage

```
tiledb_sparse(
  uri,
  query_type = c("READ", "WRITE"),
  as.data.frame = FALSE,
  attrs = character(),
  extended = TRUE,
  ctx = tiledb_get_context()
)
```

### Arguments

uri	uri path to the tiledb dense array
query_type	optionally loads the array in "READ" or "WRITE" only modes.
as.data.frame	optional logical switch, defaults to "FALSE"
attrs	optional character vector to select attributes, default is empty implying all are selected
extended	optional logical switch selecting wide 'data.frame' format, defaults to "TRUE"
ctx	tiledb_ctx (optional)

**Value**

tiledb\_sparse array object

---

tiledb_sparse-class	<i>An S4 class for a TileDB sparse array</i>
---------------------	--

---

**Description**

An S4 class for a TileDB sparse array

**Slots**

ctx A TileDB context object  
uri A character description  
as.data.frame A logical value  
attrs A character vector  
extended A logical value  
ptr External pointer to the underlying implementation

---

tiledb_stats_disable	<i>Disable stats counters</i>
----------------------	-------------------------------

---

**Description**

Disable stats counters

**Usage**

tiledb\_stats\_disable()

---

tiledb_stats_dump	<i>Dump stats to file</i>
-------------------	---------------------------

---

## Description

Dump stats to file

## Usage

```
tiledb_stats_dump(path)
```

## Arguments

path	to stats file
------	---------------

## Examples

```
pth <- tempfile()
tiledb_stats_dump(pth)
cat(readLines(pth)[1:10], sep = "\n")
```

---

tiledb_stats_enable	<i>Enable stats counters</i>
---------------------	------------------------------

---

## Description

Enable stats counters

## Usage

```
tiledb_stats_enable()
```

---

tiledb_subarray	<i>Query a array using a subarray vector</i>
-----------------	--

---

**Description**

tiledb\_subarray returns a results of query

**Usage**

```
tiledb_subarray(A, subarray_vector, attrs = c())
```

**Arguments**

A	tiledb_sparse or tiledb_dense
subarray_vector	subarray to query
attrs	list of attributes to query

**Value**

list of attributes being returned with query results

---

tiledb_version	<i>The version of the libtiledb library</i>
----------------	---

---

**Description**

The version of the libtiledb library

**Usage**

```
tiledb_version(compact = FALSE)
```

**Arguments**

compact	Logical value indicating wheter a compact package_version object should be returned
---------	---

**Value**

An named int vector c(major, minor, patch), or if select, a package\_version object

**Examples**

```
tiledb_version()
tiledb_version(compact = TRUE)
```

---

tiledb_vfs	<i>Creates a tiledb_vfs object</i>
------------	------------------------------------

---

### Description

Creates a tiledb\_vfs object

### Usage

```
tiledb_vfs(config = NULL, ctx = tiledb_get_context())
```

### Arguments

config	(optional) character vector of config parameter names, values
ctx	(optional) A TileDB Ctx object

### Value

tiledb\_vfs object

### Examples

```
# default configuration
vfs <- tiledb_vfs()
```

---

tiledb_vfs-class	<i>An S4 class for a TileDB VFS object</i>
------------------	--

---

### Description

An S4 class for a TileDB VFS object

### Slots

ptr An external pointer to the underlying implementation

---

tiledb_vfs_create_bucket
<i>Create a VFS Bucket</i>

---

**Description**

Create a VFS Bucket

**Usage**

tiledb\_vfs\_create\_bucket(vfs, uri)

**Arguments**

- |     |   |
|-----|---|
| vfs | TileDB VFS object                                       |
| uri | Character variable with a URI describing a cloud bucket |

**Value**

The uri value

---

tiledb_vfs_create_dir	<i>Create a VFS Directory</i>
-----------------------	-------------------------------

---

**Description**

Create a VFS Directory

**Usage**

tiledb\_vfs\_create\_dir(vfs, uri)

**Arguments**

- |     |  |
|-----|--|
| vfs | TileDB VFS object  |
| uri | Character variable with a URI describing a diretory path |

**Value**

The uri value of the created directory



---

tiledb\_vfs\_empty\_bucket      *Empty a VFS Bucket*

---

### Description

Empty a VFS Bucket

### Usage

tiledb\_vfs\_empty\_bucket(vfs, uri)

### Arguments

vfs	TileDB VFS object
uri	Character variable with a URI describing a cloud bucket

### Value

The URI value that was emptied

---

tiledb\_vfs\_file\_size      *Return VFS File Size*

---

### Description

Return VFS File Size

### Usage

tiledb\_vfs\_file\_size(vfs, uri)

### Arguments

vfs	TileDB VFS object
uri	Character variable with a URI describing a file path

### Value

The size removed file

---

tiledb\_vfs\_is\_bucket     *Check for VFS Bucket*

---

**Description**

Check for VFS Bucket

**Usage**

```
tiledb_vfs_is_bucket(vfs, uri)
```

**Arguments**

vfs	TileDB VFS object
uri	Character variable with a URI describing a cloud bucket

**Value**

A boolean value indicating if it is a valid bucket

**Examples**

```
## Not run:
cfg <- tiledb_config()
cfg["vfs.s3.region"] <- "us-west-1"
ctx <- tiledb_ctx(cfg)
vfs <- tiledb_vfs()
tiledb_vfs_is_bucket(vfs, "s3://tiledb-public-us-west-1/test-array-4x4")

## End(Not run)
```

---

tiledb\_vfs\_is\_dir     *Test for VFS Directory*

---

**Description**

Test for VFS Directory

**Usage**

```
tiledb_vfs_is_dir(vfs, uri)
```

**Arguments**

vfs	TileDB VFS object
uri	Character variable with a URI describing a directory path

**Value**

A boolean value indicating if it is a directory

---

tiledb_vfs_is_empty_bucket
<i>Check for empty VFS Bucket</i>

---

**Description**

Check for empty VFS Bucket

**Usage**

tiledb\_vfs\_is\_empty\_bucket(vfs, uri)

**Arguments**

vfs	TileDB VFS object
uri	Character variable with a URI describing a cloud bucket

**Value**

A boolean value indicating if it is an empty bucket

**Examples**

```
## Not run:
cfg <- tiledb_config()
cfg["vfs.s3.region"] <- "us-west-1"
ctx <- tiledb_ctx(cfg)
vfs <- tiledb_vfs()
tiledb_vfs_is_empty_bucket(vfs, "s3://tiledb-public-us-west-1/test-array-4x4")

## End(Not run)
```

---

tiledb_vfs_is_file	<i>Test for VFS File</i>
--------------------	--------------------------

---

**Description**

Test for VFS File

**Usage**

```
tiledb_vfs_is_file(vfs, uri)
```

**Arguments**

vfs	TileDB VFS object
uri	Character variable with a URI describing a file path

**Value**

A boolean value indicating if it is a file

---

tiledb_vfs_move_dir	<i>Move (or rename) a VFS Directory</i>
---------------------	---

---

**Description**

Move (or rename) a VFS Directory

**Usage**

```
tiledb_vfs_move_dir(vfs, olduri, newuri)
```

**Arguments**

vfs	TileDB VFS object
olduri	Character variable with an existing URI describing a directory path
newuri	Character variable with a new desired URI directory path

**Value**

The newuri value of the moved directory

---

tiledb\_vfs\_move\_file    *Move (or rename) a VFS File*

---

### Description

Move (or rename) a VFS File

### Usage

```
tiledb_vfs_move_file(vfs, olduri, newuri)
```

### Arguments

vfs	TileDB VFS object
olduri	Character variable with an existing URI describing a file path
newuri	Character variable with a new desired URI file path

### Value

The newuri value of the moved file

---

tiledb\_vfs\_remove\_bucket  
*Remove a VFS Bucket*

---

### Description

Remove a VFS Bucket

### Usage

```
tiledb_vfs_remove_bucket(vfs, uri)
```

### Arguments

vfs	TileDB VFS object
uri	Character variable with a URI describing a cloud bucket

### Value

The uri value

---

`tiledb_vfs_remove_dir` *Remove a VFS Directory*

---

**Description**

Remove a VFS Directory

**Usage**

```
tiledb_vfs_remove_dir(vfs, uri)
```

**Arguments**

<code>vfs</code>	TileDB VFS object
<code>uri</code>	Character variable with a URI describing a directory path

**Value**

The uri value of the removed directory

---

`tiledb_vfs_remove_file`  
*Remove a VFS File*

---

**Description**

Remove a VFS File

**Usage**

```
tiledb_vfs_remove_file(vfs, uri)
```

**Arguments**

<code>vfs</code>	TileDB VFS object
<code>uri</code>	Character variable with a URI describing a file path

**Value**

The uri value of the removed file

---

tiledb_vfs_touch	<i>Touch a VFS URI Resource</i>
------------------	---------------------------------

---

**Description**

Touch a VFS URI Resource

**Usage**

tiledb\_vfs\_touch(vfs, uri)

**Arguments**

- vfs                   TileDB VFS object
- uri                   Character variable with a URI describing a bucket, file or directory

**Value**

The uri value

---

tile_order,tiledb_array_schema-method
<i>Returns the tile layout string associated with the tiledb_array_schema</i>

---

**Description**

Returns the tile layout string associated with the tiledb\_array\_schema

**Usage**

```
## S4 method for signature 'tiledb_array_schema'  
tile_order(object)
```

**Arguments**

- object               tiledb object

---

```
[, tiledb_array, ANY-method
```

*Returns a TileDB array, allowing for specific subset ranges.*

---

### Description

Heterogenous domains are supported, including timestamps and characters.

### Usage

```
## S4 method for signature 'tiledb_array, ANY'
x[i, j, ..., drop = FALSE]
```

### Arguments

x	tiledb_array object
i	optional row index expression which can be a list in which case minimum and maximum of each list element determine a range; multiple list elements can be used to supply multiple ranges.
j	optional column index expression which can be a list in which case minimum and maximum of each list element determine a range; multiple list elements can be used to supply multiple ranges.
...	Extra parameters for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

### Details

This function may still change; the current implementation should be considered as an initial draft.

### Value

An element from the sparse array

---

```
[, tiledb_config, ANY-method
```

*Gets a config parameter value*

---

### Description

Gets a config parameter value



**Usage**

```
## S4 method for signature 'tiledb_config,ANY'
x[i, j, ..., drop = FALSE]
```

**Arguments**

x	tiledb_config object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

**Value**

a config string value if parameter exists, else NA

**Examples**

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]
cfg["does_not_exist"]
```

---

```
[,tiledb_dense,ANY-method
```

*Gets a dense array value*

---

**Description**

Gets a dense array value

**Usage**

```
## S4 method for signature 'tiledb_dense,ANY'
x[i, j, ..., drop = FALSE]
```

**Arguments**

x	dense array object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

**Value**

An element from a dense array

---

```
[,tiledb_filter_list,ANY-method
```

*Returns the filter at given index*

---

### Description

Returns the filter at given index

### Usage

```
## S4 method for signature 'tiledb_filter_list,ANY'
x[i, j, ..., drop = FALSE]
```

### Arguments

x	tiledb_config object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default false.

### Value

object tiledb\_filter

### Examples

```
flt <- tiledb_filter("ZSTD")
tiledb_filter_set_option(flt, "COMPRESSION_LEVEL", 5)
filter_list <- tiledb_filter_list(c(flt))
filter_list[0]
```

---

```
[,tiledb_sparse,ANY-method
```

*Gets a sparse array value*

---

### Description

Gets a sparse array value

### Usage

```
## S4 method for signature 'tiledb_sparse,ANY'
x[i, j, ..., drop = FALSE]
```

**Arguments**

x	sparse array object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
drop	Optional logical switch to drop dimensions, default FALSE, currently unused.

**Value**

An element from the sparse array

---

```
[<-, tiledb_array, ANY, ANY, ANY-method
```

*Sets a tiledb array value or value range*

---

**Description**

This function assigns a right-hand side object, typically a data.frame or something that can be coerced to a data.frame, to a tiledb array.

**Usage**

```
## S4 replacement method for signature 'tiledb_array, ANY, ANY, ANY'
x[i, j, ...] <- value
```

**Arguments**

x	sparse or dense TileDB array object
i	parameter row index
j	parameter column index
...	Extra parameter for method signature, currently unused.
value	The value being assigned

**Details**

For sparse matrices, row and column indices can either be supplied as part of the left-hand side object, or as part of the data.frame provided appropriate column names.

This function may still change; the current implementation should be considered as an initial draft.

**Value**

The modified object

**Examples**

```
## Not run:
uri <- "quickstart_sparse"      ## as created by the other example
arr <- tiledb_array(uri)       ## open array
df <- arr[]                    ## read current content
## First approach: matching data.frame with appropriate row and column
newdf <- data.frame(rows=c(1,2,2), cols=c(1,3,4), a=df$a+100)
## Second approach: supply indices explicitly
arr[c(1,2), c(1,3)] <- c(42,43) ## two values
arr[2, 4] <- 88                ## or just one

## End(Not run)
```

---

```
[<-, tiledb_config, ANY, ANY, ANY-method
      Sets a config parameter value
```

---

**Description**

Sets a config parameter value

**Usage**

```
## S4 replacement method for signature 'tiledb_config, ANY, ANY, ANY'
x[i, j] <- value
```

**Arguments**

x	tiledb_config object
i	parameter key string
j	parameter key string
value	value to set, will be converted into a string

**Value**

updated tiledb\_config object

**Examples**

```
cfg <- tiledb_config()
cfg["sm.tile_cache_size"]

# set tile cache size to custom value
cfg["sm.tile_cache_size"] <- 100
cfg["sm.tile_cache_size"]
```

---

```
[<-, tiledb_dense, ANY, ANY, ANY-method
```

*Sets a dense array value*

---

**Description**

Sets a dense array value

**Usage**

```
## S4 replacement method for signature 'tiledb_dense, ANY, ANY, ANY'
x[i, j, ...] <- value
```

**Arguments**

x	dense array object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
value	The value being assigned

**Value**

The modified object

---

```
[<-, tiledb_sparse, ANY, ANY, ANY-method
```

*Sets a sparse array value*

---

**Description**

Sets a sparse array value

**Usage**

```
## S4 replacement method for signature 'tiledb_sparse, ANY, ANY, ANY'
x[i, j, ...] <- value
```

**Arguments**

x	sparse array object
i	parameter key string
j	parameter key string, currently unused.
...	Extra parameter for method signature, currently unused.
value	The value being assigned

**Value**

The modified object

# Index

```

* stats
  tiledb_stats_disable, 84
  tiledb_stats_dump, 85
  tiledb_stats_enable, 85
[, tiledb_array
  ([, tiledb_array, ANY-method), 96
[, tiledb_array, ANY, ANY, tiledb_array-method
  ([, tiledb_array, ANY-method), 96
[, tiledb_array, ANY, tiledb_array-method
  ([, tiledb_array, ANY-method), 96
[, tiledb_array, ANY-method, 96
[, tiledb_array-method
  ([, tiledb_array, ANY-method), 96
[, tiledb_config
  ([, tiledb_config, ANY-method),
  96
[, tiledb_config, ANY, ANY, tiledb_config-method
  ([, tiledb_config, ANY-method),
  96
[, tiledb_config, ANY, tiledb_config-method
  ([, tiledb_config, ANY-method),
  96
[, tiledb_config, ANY-method, 96
[, tiledb_config-method
  ([, tiledb_config, ANY-method),
  96
[, tiledb_dense
  ([, tiledb_dense, ANY-method), 97
[, tiledb_dense, ANY, ANY, tiledb_dense-method
  ([, tiledb_dense, ANY-method), 97
[, tiledb_dense, ANY, tiledb_dense-method
  ([, tiledb_dense, ANY-method), 97
[, tiledb_dense, ANY-method, 97
[, tiledb_dense-method
  ([, tiledb_dense, ANY-method), 97
[, tiledb_filter_list
  ([, tiledb_filter_list, ANY-method),
  98
[, tiledb_filter_list, ANY, ANY, tiledb_filter_list-method
  ([, tiledb_filter_list, ANY-method),
  98
[, tiledb_filter_list, ANY, tiledb_filter_list-method
  ([, tiledb_filter_list, ANY-method),
  98
[, tiledb_filter_list, ANY-method, 98
[, tiledb_sparse
  ([, tiledb_sparse, ANY-method),
  98
[, tiledb_sparse, ANY, ANY, tiledb_sparse-method
  ([, tiledb_sparse, ANY-method),
  98
[, tiledb_sparse, ANY, tiledb_sparse-method
  ([, tiledb_sparse, ANY-method),
  98
[, tiledb_sparse, ANY-method, 98
[, tiledb_sparse-method
  ([, tiledb_sparse, ANY-method),
  98
[<-, tiledb_array, ANY, ANY, ANY-method,
  99
[<-, tiledb_config, ANY, ANY, ANY-method,
  100
[<-, tiledb_dense, ANY, ANY, ANY-method,
  101
[<-, tiledb_sparse, ANY, ANY, ANY-method,
  101
[<-, tiledb_array
  ([<-, tiledb_array, ANY, ANY, ANY-method),
  99
[<-, tiledb_array, ANY, ANY, tiledb_array-method
  ([<-, tiledb_array, ANY, ANY, ANY-method),
  99
[<-, tiledb_array, ANY, tiledb_array-method
  ([<-, tiledb_array, ANY, ANY, ANY-method),
  99

```

[\[<-, tiledb\\_array-method](#) [attrs \(domain\), 21](#)  
[\(\[<-, tiledb\\_array, ANY, ANY, ANY-method\), attrs, tiledb\\_array, ANY-method, 9](#)  
[99](#) [attrs, tiledb\\_array\\_schema, ANY-method,](#)  
[\[<-, tiledb\\_config](#) [9](#)  
[\(\[<-, tiledb\\_config, ANY, ANY, ANY-method\), attrs, tiledb\\_array\\_schema, character-method,](#)  
[100](#) [10](#)  
[\[<-, tiledb\\_config, ANY, ANY, tiledb\\_config-method\), attrs, tiledb\\_array\\_schema, numeric-method,](#)  
[\(\[<-, tiledb\\_config, ANY, ANY, ANY-method\),](#) [11](#)  
[100](#) [attrs, tiledb\\_dense, ANY-method, 11](#)  
[\[<-, tiledb\\_config, ANY, tiledb\\_config-method](#) [attrs, tiledb\\_sparse, ANY-method, 12](#)  
[\(\[<-, tiledb\\_config, ANY, ANY, ANY-method\), attrs<-,](#) [12](#)  
[100](#) [attrs<-, tiledb\\_array-method, 13](#)  
[\[<-, tiledb\\_config-method](#) [attrs<-, tiledb\\_sparse-method, 13](#)  
[\(\[<-, tiledb\\_config, ANY, ANY, ANY-method\), attrs<-, tiledb\\_dense-method \(attrs<-\),](#)  
[100](#) [12](#)  
[\[<-, tiledb\\_dense](#)  
[\(\[<-, tiledb\\_dense, ANY, ANY, ANY-method\), cell\\_order \(domain\), 21](#)  
[101](#) [cell\\_order, tiledb\\_array\\_schema-method,](#)  
[\[<-, tiledb\\_dense, ANY, ANY, tiledb\\_dense-method](#) [14](#)  
[\(\[<-, tiledb\\_dense, ANY, ANY, ANY-method\), cell\\_val\\_num \(domain\), 21](#)  
[101](#) [cell\\_val\\_num, tiledb\\_attr-method, 14](#)  
[\[<-, tiledb\\_dense, ANY, tiledb\\_dense-method](#) [config \(domain\), 21](#)  
[\(\[<-, tiledb\\_dense, ANY, ANY, ANY-method\), config, tiledb\\_ctx-method, 15](#)  
[101](#)  
[\[<-, tiledb\\_dense-method](#) [datatype \(domain\), 21](#)  
[\(\[<-, tiledb\\_dense, ANY, ANY, ANY-method\), datatype, tiledb\\_attr-method, 15](#)  
[101](#) [datatype, tiledb\\_dim-method, 16](#)  
[\[<-, tiledb\\_sparse](#) [datatype, tiledb\\_domain-method, 17](#)  
[\(\[<-, tiledb\\_sparse, ANY, ANY, ANY-method\), dim. tiledb\\_array\\_schema, 17](#)  
[101](#) [dim. tiledb\\_dim, 18](#)  
[\[<-, tiledb\\_sparse, ANY, ANY, tiledb\\_sparse-method\), dim. tiledb\\_domain, 19](#)  
[\(\[<-, tiledb\\_sparse, ANY, ANY, ANY-method\), dimensions \(domain\), 21](#)  
[101](#) [dimensions, tiledb\\_array\\_schema-method,](#)  
[\[<-, tiledb\\_sparse, ANY, tiledb\\_sparse-method](#) [19](#)  
[\(\[<-, tiledb\\_sparse, ANY, ANY, ANY-method\), dimensions, tiledb\\_domain-method, 20](#)  
[101](#) [domain, 21](#)  
[\[<-, tiledb\\_sparse-method](#) [domain, tiledb\\_array\\_schema-method, 22](#)  
[\(\[<-, tiledb\\_sparse, ANY, ANY, ANY-method\), domain, tiledb\\_dim-method, 22](#)  
[101](#)  
[extended, 23](#)  
[allows\\_dups, 6](#) [extended, tiledb\\_array-method](#)  
[allows\\_dups, tiledb\\_array\\_schema-method](#) [\(extended\), 23](#)  
[\(allows\\_dups\), 6](#) [extended<-, 23](#)  
[allows\\_dups<-, 6](#) [extended<-, tiledb\\_array-method](#)  
[allows\\_dups<-, tiledb\\_array\\_schema-method](#) [\(extended<-\), 23](#)  
[\(allows\\_dups<-\), 6](#)  
[as.data.frame.tiledb\\_config, 7](#) [filter\\_list \(domain\), 21](#)  
[as.vector.tiledb\\_config, 7](#) [filter\\_list, tiledb\\_array\\_schema-method,](#)  
[as\\_data\\_frame, 8](#) [24](#)



[filter\\_list](#), [tiledb\\_attr](#)-method, [24](#)  
[fromDataFrame](#), [25](#)  
  
[generics](#) (domain), [21](#)  
  
[is.anonymous](#), [26](#)  
[is.anonymous.tiledb\\_dim](#), [27](#)  
[is.integral](#) (domain), [21](#)  
[is.integral](#), [tiledb\\_domain](#)-method, [27](#)  
[is.sparse](#) (domain), [21](#)  
[is.sparse](#), [tiledb\\_array\\_schema](#)-method, [28](#)  
[is.sparse](#), [tiledb\\_dense](#)-method, [28](#)  
[is.sparse](#), [tiledb\\_sparse](#)-method, [29](#)  
  
[limitTileDBCores](#), [29](#)  
  
[max\\_chunk\\_size](#) (domain), [21](#)  
[max\\_chunk\\_size](#), [tiledb\\_filter\\_list](#)-method, [30](#)  
  
[name](#) (domain), [21](#)  
[name](#), [tiledb\\_attr](#)-method, [31](#)  
[name](#), [tiledb\\_dim](#)-method, [31](#)  
[nfilters](#) (domain), [21](#)  
[nfilters](#), [tiledb\\_filter\\_list](#)-method, [32](#)  
  
[print.tiledb\\_metadata](#), [33](#)  
  
[r\\_to\\_tiledb\\_type](#), [36](#)  
[return.data.frame](#), [33](#)  
[return.data.frame](#), [tiledb\\_array](#)-method, [34](#)  
[return.data.frame](#), [tiledb\\_dense](#)-method (return.data.frame), [33](#)  
[return.data.frame](#), [tiledb\\_sparse](#)-method, [34](#)  
[return.data.frame<-](#), [35](#)  
[return.data.frame<-](#), [tiledb\\_array](#)-method, [35](#)  
[return.data.frame<-](#), [tiledb\\_sparse](#)-method, [36](#)  
[return.data.frame<-](#), [tiledb\\_dense](#)-method (return.data.frame<-), [35](#)  
  
[schema](#) (domain), [21](#)  
[schema](#), [tiledb\\_array](#)-method, [37](#)  
[schema](#), [tiledb\\_dense](#)-method, [37](#)  
[schema](#), [tiledb\\_sparse](#)-method, [38](#)  
[selected\\_ranges](#), [38](#)  
  
[selected\\_ranges](#), [tiledb\\_array](#)-method (selected\_ranges), [38](#)  
[selected\\_ranges<-](#), [39](#)  
[selected\\_ranges<-](#), [tiledb\\_array](#)-method (selected\_ranges<-), [39](#)  
[set\\_max\\_chunk\\_size](#) (domain), [21](#)  
[set\\_max\\_chunk\\_size](#), [tiledb\\_filter\\_list](#), [numeric](#)-method, [39](#)  
  
[show](#), [tiledb\\_array](#)-method, [40](#)  
[show](#), [tiledb\\_array\\_schema](#)-method, [40](#)  
[show](#), [tiledb\\_attr](#)-method, [41](#)  
[show](#), [tiledb\\_config](#)-method, [41](#)  
[show](#), [tiledb\\_dense](#)-method, [42](#)  
[show](#), [tiledb\\_domain](#)-method, [42](#)  
[show](#), [tiledb\\_sparse](#)-method, [43](#)  
  
[tile](#) (domain), [21](#)  
[tile](#), [tiledb\\_dim](#)-method, [43](#)  
[tile\\_order](#) (domain), [21](#)  
[tile\\_order](#), [tiledb\\_array\\_schema](#)-method, [95](#)  
  
[tiledb](#)-package, [5](#)  
[tiledb\\_array](#), [44](#)  
[tiledb\\_array](#)-class, [45](#)  
[tiledb\\_array\\_close](#), [45](#)  
[tiledb\\_array\\_create](#), [46](#)  
[tiledb\\_array\\_is\\_heterogeneous](#), [46](#)  
[tiledb\\_array\\_is\\_homogeneous](#), [47](#)  
[tiledb\\_array\\_open](#), [47](#)  
[tiledb\\_array\\_schema](#), [48](#)  
[tiledb\\_array\\_schema](#)-class, [49](#)  
[tiledb\\_attr](#), [49](#)  
[tiledb\\_attr](#)-class, [50](#)  
[tiledb\\_config](#), [50](#)  
[tiledb\\_config](#)-class, [51](#)  
[tiledb\\_config\\_load](#), [51](#)  
[tiledb\\_config\\_save](#), [52](#)  
[tiledb\\_ctx](#), [52](#)  
[tiledb\\_ctx](#)-class, [53](#)  
[tiledb\\_ctx\\_set\\_default\\_tags](#), [53](#)  
[tiledb\\_ctx\\_set\\_tag](#), [54](#)  
[tiledb\\_delete\\_metadata](#), [54](#)  
[tiledb\\_dense](#), [45](#), [55](#)  
[tiledb\\_dense](#)-class, [55](#)  
[tiledb\\_dim](#), [56](#)  
[tiledb\\_dim](#)-class, [57](#)  
[tiledb\\_domain](#), [57](#)  
[tiledb\\_domain](#)-class, [57](#)  
[tiledb\\_filter](#), [58](#)

tiledb\_filter-class, [59](#)  
 tiledb\_filter\_get\_option, [59](#)  
 tiledb\_filter\_list, [60](#)  
 tiledb\_filter\_list-class, [60](#)  
 tiledb\_filter\_set\_option, [61](#)  
 tiledb\_filter\_type, [61](#)  
 tiledb\_get\_all\_metadata, [62](#)  
 tiledb\_get\_context, [62](#)  
 tiledb\_get\_metadata, [63](#)  
 tiledb\_group\_create, [63](#)  
 tiledb\_has\_metadata, [64](#)  
 tiledb\_is\_supported\_fs, [64](#)  
 tiledb\_ndim (domain), [21](#)  
 tiledb\_ndim, tiledb\_array\_schema-method, [65](#)  
 tiledb\_ndim, tiledb\_dim-method, [66](#)  
 tiledb\_ndim, tiledb\_domain-method, [66](#)  
 tiledb\_num\_metadata, [67](#)  
 tiledb\_object\_ls, [67](#)  
 tiledb\_object\_mv, [68](#)  
 tiledb\_object\_rm, [68](#)  
 tiledb\_object\_type, [69](#)  
 tiledb\_object\_walk, [69](#)  
 tiledb\_put\_metadata, [70](#)  
 tiledb\_query, [70](#)  
 tiledb\_query-class, [71](#)  
 tiledb\_query\_add\_range, [71](#)  
 tiledb\_query\_add\_range\_with\_type, [72](#)  
 tiledb\_query\_alloc\_buffer\_ptr\_char, [72](#)  
 tiledb\_query\_alloc\_buffer\_ptr\_char\_subarray, [73](#)  
 tiledb\_query\_buffer\_alloc\_ptr, [74](#)  
 tiledb\_query\_create\_buffer\_ptr, [74](#)  
 tiledb\_query\_create\_buffer\_ptr\_char, [75](#)  
 tiledb\_query\_finalize, [75](#)  
 tiledb\_query\_get\_buffer\_char, [76](#)  
 tiledb\_query\_get\_buffer\_ptr, [76](#)  
 tiledb\_query\_get\_layout, [77](#)  
 tiledb\_query\_result\_buffer\_elements, [77](#)  
 tiledb\_query\_set\_buffer, [78](#)  
 tiledb\_query\_set\_buffer\_ptr, [78](#)  
 tiledb\_query\_set\_buffer\_ptr\_char, [79](#)  
 tiledb\_query\_set\_layout, [79](#)  
 tiledb\_query\_set\_subarray, [80](#)  
 tiledb\_query\_status, [80](#)  
 tiledb\_query\_submit, [81](#)  
 tiledb\_query\_type, [81](#)  
 tiledb\_schema\_get\_names, [82](#)  
 tiledb\_schema\_get\_types, [82](#)  
 tiledb\_set\_context, [83](#)  
 tiledb\_sparse, [45](#), [83](#)  
 tiledb\_sparse-class, [84](#)  
 tiledb\_stats\_disable, [84](#)  
 tiledb\_stats\_dump, [85](#)  
 tiledb\_stats\_enable, [85](#)  
 tiledb\_subarray, [86](#)  
 tiledb\_version, [86](#)  
 tiledb\_vfs, [87](#)  
 tiledb\_vfs-class, [87](#)  
 tiledb\_vfs\_create\_bucket, [88](#)  
 tiledb\_vfs\_create\_dir, [88](#)  
 tiledb\_vfs\_empty\_bucket, [89](#)  
 tiledb\_vfs\_file\_size, [89](#)  
 tiledb\_vfs\_is\_bucket, [90](#)  
 tiledb\_vfs\_is\_dir, [90](#)  
 tiledb\_vfs\_is\_empty\_bucket, [91](#)  
 tiledb\_vfs\_is\_file, [92](#)  
 tiledb\_vfs\_move\_dir, [92](#)  
 tiledb\_vfs\_move\_file, [93](#)  
 tiledb\_vfs\_remove\_bucket, [93](#)  
 tiledb\_vfs\_remove\_dir, [94](#)  
 tiledb\_vfs\_remove\_file, [94](#)  
 tiledb\_vfs\_touch, [95](#)