

# Package ‘tiledb’

April 16, 2024

**Type** Package

**Version** 0.26.0

**Title** Modern Database Engine for Complex Data Based on Multi-Dimensional Arrays

**Description** The modern database 'TileDB' introduces a powerful on-disk format for storing and accessing any complex data based on 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. This package provides the R support.

**Copyright** TileDB, Inc.

**License** MIT + file LICENSE

**URL** <https://github.com/TileDB-Inc/TileDB-R>,  
<https://tiledb-inc.github.io/TileDB-R/>

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

**SystemRequirements** A C++17 compiler is required; on macOS compilation version 11.0 or later is required. Optionally cmake (only when TileDB source build selected), curl (only when TileDB source build selected), and git (only when TileDB source build selected); on x86\_64 and M1 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 (>= 1.0.8), nanotime, spd, nanoarrow

**LinkingTo** Rcpp, RcppInt64, nanoarrow

**Suggests** tinytest, simplermardown, curl, bit64, Matrix,  
palmerpenguins, nycflights13, data.table, tibble, arrow

**VignetteBuilder** simplermardown

**RoxygenNote** 7.3.1

**Encoding** UTF-8

**NeedsCompilation** yes

**Author** TileDB, Inc. [aut, cph],  
Dirk Eddelbuettel [cre]

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

**Repository** CRAN

**Date/Publication** 2024-04-16 02:40:03 UTC

## R topics documented:

allows_dups	9
allows_dups<-	10
array Consolidate	10
array Vacuum	11
as.data.frame.tiledb_config	12
as.vector.tiledb_config	12
attrs, tiledb_array, ANY-method	13
attrs, tiledb_array_schema, ANY-method	13
attrs, tiledb_array_schema, character-method	14
attrs, tiledb_array_schema, numeric-method	15
attrs<-, tiledb_array-method	16
capacity	16
capacity<-	17
cell_order, tiledb_array_schema-method	17
cell_val_num	18
cell_val_num, tiledb_dim-method	18
cell_val_num<-	19
completedBatched	19
config, tiledb_ctx-method	20
createBatched	21
datatype, tiledb_attr-method	21
datatype, tiledb_dim-method	22
datatype, tiledb_domain-method	22
datetimes_as_int64	23
datetimes_as_int64<-	24
describe	24
dim.tiledb_array_schema	25
dim.tiledb_dim	25
dim.tiledb_domain	26
dimensions, tiledb_array_schema-method	27
dimensions, tiledb_domain-method	27
domain, tiledb_array_schema-method	28
domain, tiledb_dim-method	29
extended	29
extended<-	30
fetchBatched	30

filter_list,tiledb_array_schema-method . . . . .	31
filter_list,tiledb_attr-method . . . . .	31
filter_list,tiledb_dim-method . . . . .	32
filter_list<-,tiledb_attr-method . . . . .	32
filter_list<-,tiledb_dim-method . . . . .	33
fromDataFrame . . . . .	33
fromMatrix . . . . .	35
fromSparseMatrix . . . . .	36
generics . . . . .	37
has_attribute . . . . .	38
is.anonymous . . . . .	39
is.anonymous.tiledb_dim . . . . .	40
is.integral,tiledb_domain-method . . . . .	40
is.sparse,tiledb_array_schema-method . . . . .	41
limitTileDBCores . . . . .	41
max_chunk_size . . . . .	42
name,tiledb_attr-method . . . . .	43
name,tiledb_dim-method . . . . .	44
nfilters,tiledb_filter_list-method . . . . .	44
parse_query_condition . . . . .	45
print.tiledb_metadata . . . . .	46
query_condition . . . . .	47
query_condition<- . . . . .	47
query_layout . . . . .	48
query_layout<- . . . . .	48
query_statistics . . . . .	49
query_statistics<- . . . . .	49
raw_dump,tiledb_array_schema-method . . . . .	50
raw_dump,tiledb_attr-method . . . . .	50
raw_dump,tiledb_domain-method . . . . .	51
return.array . . . . .	51
return.array<- . . . . .	52
return.data.frame,tiledb_array-method . . . . .	52
return.data.frame<-,tiledb_array-method . . . . .	53
return.matrix . . . . .	53
return.matrix<- . . . . .	54
return_as . . . . .	54
return_as<- . . . . .	55
r_to_tiledb_type . . . . .	55
save_allocation_size_preference . . . . .	56
save_return_as_preference . . . . .	57
schema,character-method . . . . .	58
schema,tiledb_array-method . . . . .	58
schema_check . . . . .	59
selected_points . . . . .	59
selected_points<- . . . . .	60
selected_ranges . . . . .	60
selected_ranges<- . . . . .	61

set_max_chunk_size . . . . .	62
show,tiledb_array-method . . . . .	62
show,tiledb_array_schema-method . . . . .	63
show,tiledb_attr-method . . . . .	63
show,tiledb_config-method . . . . .	64
show,tiledb_dim-method . . . . .	64
show,tiledb_domain-method . . . . .	65
show,tiledb_filter-method . . . . .	65
show,tiledb_filter_list-method . . . . .	66
show,tiledb_group-method . . . . .	66
statusBatched . . . . .	67
strings_as_factors . . . . .	67
strings_as_factors<- . . . . .	68
tdb_collect,tiledb_array-method . . . . .	68
tdb_filter,tiledb_array-method . . . . .	69
tdb_select,tiledb_array-method . . . . .	69
tile,tiledb_dim-method . . . . .	70
tiledb_array . . . . .	70
tiledb_array-class . . . . .	73
tiledb_array_apply_aggregate . . . . .	74
tiledb_array_close . . . . .	74
tiledb_array_create . . . . .	75
tiledb_array_delete_fragments . . . . .	75
tiledb_array_get_non_empty_domain_from_index . . . . .	76
tiledb_array_get_non_empty_domain_from_name . . . . .	77
tiledb_array_has_enumeration . . . . .	77
tiledb_array_is_heterogeneous . . . . .	78
tiledb_array_is_homogeneous . . . . .	78
tiledb_array_is_open . . . . .	79
tiledb_array_open . . . . .	79
tiledb_array_open_at . . . . .	80
tiledb_array_schema . . . . .	80
tiledb_array_schema-class . . . . .	81
tiledb_array_schema_evolution . . . . .	82
tiledb_array_schema_evolution-class . . . . .	82
tiledb_array_schema_evolution_add_attribute . . . . .	83
tiledb_array_schema_evolution_add_enumeration . . . . .	83
tiledb_array_schema_evolution_add_enumeration_empty . . . . .	84
tiledb_array_schema_evolution_array_evolve . . . . .	84
tiledb_array_schema_evolution_drop_attribute . . . . .	85
tiledb_array_schema_evolution_drop_enumeration . . . . .	85
tiledb_array_schema_evolution_extend_enumeration . . . . .	86
tiledb_array_schema_set_coords_filter_list . . . . .	87
tiledb_array_schema_set_enumeration_empty . . . . .	87
tiledb_array_schema_set_offsets_filter_list . . . . .	88
tiledb_array_schema_set_validity_filter_list . . . . .	89
tiledb_array_schema_version . . . . .	89
tiledb_array_upgrade_version . . . . .	90

<code>tiledb_arrow_array_ptr</code>	90
<code>tiledb_attr</code>	91
<code>tiledb_attr-class</code>	92
<code>tiledb_attribute_get_cell_size</code>	92
<code>tiledb_attribute_get_enumeration</code>	92
<code>tiledb_attribute_get_fill_value</code>	93
<code>tiledb_attribute_get_nullable</code>	93
<code>tiledb_attribute_has_enumeration</code>	94
<code>tiledb_attribute_is_ordered_enumeration_ptr</code>	94
<code>tiledb_attribute_is_variable_sized</code>	95
<code>tiledb_attribute_set_enumeration_name</code>	95
<code>tiledb_attribute_set_fill_value</code>	96
<code>tiledb_attribute_set_nullable</code>	96
<code>tiledb_config</code>	97
<code>tiledb_config-class</code>	97
<code>tiledb_config_as_built_json</code>	98
<code>tiledb_config_as_built_show</code>	98
<code>tiledb_config_load</code>	99
<code>tiledb_config_save</code>	99
<code>tiledb_config_unset</code>	100
<code>tiledb_ctx</code>	100
<code>tiledb_ctx-class</code>	101
<code>tiledb_ctx_set_default_tags</code>	101
<code>tiledb_ctx_set_tag</code>	102
<code>tiledb_ctx_stats</code>	102
<code>tiledb_datatype_R_type</code>	103
<code>tiledb_delete_metadata</code>	103
<code>tiledb_dim</code>	104
<code>tiledb_dim-class</code>	105
<code>tiledb_domain</code>	105
<code>tiledb_domain-class</code>	105
<code>tiledb_domain_get_dimension_from_index</code>	106
<code>tiledb_domain_get_dimension_from_name</code>	106
<code>tiledb_domain_has_dimension</code>	107
<code>tiledb_error_message</code>	107
<code>tiledb_filestore_buffer_export</code>	108
<code>tiledb_filestore_buffer_import</code>	108
<code>tiledb_filestore_schema_create</code>	109
<code>tiledb_filestore_size</code>	110
<code>tiledb_filestore_uri_export</code>	110
<code>tiledb_filestore_uri_import</code>	111
<code>tiledb_filter</code>	111
<code>tiledb_filter-class</code>	112
<code>tiledb_filter_get_option</code>	113
<code>tiledb_filter_list</code>	113
<code>tiledb_filter_list-class</code>	114
<code>tiledb_filter_set_option</code>	114
<code>tiledb_filter_type</code>	115

<code>tiledb_fragment_info</code>	115
<code>tiledb_fragment_info-class</code>	116
<code>tiledb_fragment_info_dense</code>	116
<code>tiledb_fragment_info_dump</code>	117
<code>tiledb_fragment_info_get_cell_num</code>	117
<code>tiledb_fragment_info_get_non_empty_domain_index</code>	118
<code>tiledb_fragment_info_get_non_empty_domain_name</code>	118
<code>tiledb_fragment_info_get_non_empty_domain_var_index</code>	119
<code>tiledb_fragment_info_get_non_empty_domain_var_name</code>	119
<code>tiledb_fragment_info_get_num</code>	120
<code>tiledb_fragment_info_get_size</code>	120
<code>tiledb_fragment_info_get_timestamp_range</code>	121
<code>tiledb_fragment_info_get_to_vacuum_num</code>	121
<code>tiledb_fragment_info_get_to_vacuum_uri</code>	122
<code>tiledb_fragment_info_get_unconsolidated_metadata_num</code>	122
<code>tiledb_fragment_info_get_version</code>	123
<code>tiledb_fragment_info_has_consolidated_metadata</code>	123
<code>tiledb_fragment_info_sparse</code>	124
<code>tiledb_fragment_info_uri</code>	124
<code>tiledb_get_all_metadata</code>	125
<code>tiledb_get_context</code>	125
<code>tiledb_get_metadata</code>	126
<code>tiledb_get_query_status</code>	126
<code>tiledb_get_vfs</code>	127
<code>tiledb_group</code>	127
<code>tiledb_group-class</code>	128
<code>tiledb_group_add_member</code>	128
<code>tiledb_group_close</code>	129
<code>tiledb_group_create</code>	129
<code>tiledb_group_delete_metadata</code>	130
<code>tiledb_group_get_all_metadata</code>	130
<code>tiledb_group_get_config</code>	131
<code>tiledb_group_get_metadata</code>	131
<code>tiledb_group_get_metadata_from_index</code>	132
<code>tiledb_group_has_metadata</code>	132
<code>tiledb_group_is_open</code>	133
<code>tiledb_group_is_relative</code>	133
<code>tiledb_group_member</code>	134
<code>tiledb_group_member_count</code>	134
<code>tiledb_group_member_dump</code>	135
<code>tiledb_group_metadata_num</code>	135
<code>tiledb_group_open</code>	136
<code>tiledb_group_put_metadata</code>	136
<code>tiledb_group_query_type</code>	137
<code>tiledb_group_remove_member</code>	137
<code>tiledb_group_set_config</code>	138
<code>tiledb_group_uri</code>	138
<code>tiledb_has_metadata</code>	139

tiledb_is_supported_fs . . . . .	139
tiledb_ndim,tiledb_array_schema-method . . . . .	140
tiledb_ndim,tiledb_dim-method . . . . .	141
tiledb_ndim,tiledb_domain-method . . . . .	141
tiledb_num_metadata . . . . .	142
tiledb_object_ls . . . . .	142
tiledb_object_mv . . . . .	143
tiledb_object_rm . . . . .	143
tiledb_object_type . . . . .	144
tiledb_object_walk . . . . .	144
tiledb_put_metadata . . . . .	145
tiledb_query . . . . .	145
tiledb_query-class . . . . .	146
tiledb_query_add_range . . . . .	146
tiledb_query_add_range_with_type . . . . .	147
tiledb_query_alloc_buffer_ptr_char . . . . .	147
tiledb_query_apply_aggregate . . . . .	148
tiledb_query_buffer_alloc_ptr . . . . .	149
tiledb_query_condition . . . . .	149
tiledb_query_condition-class . . . . .	150
tiledb_query_condition_combine . . . . .	150
tiledb_query_condition_create . . . . .	151
tiledb_query_condition_init . . . . .	151
tiledb_query_condition_set_use_enumeration . . . . .	152
tiledb_query_create_buffer_ptr . . . . .	153
tiledb_query_create_buffer_ptr_char . . . . .	153
tiledb_query_ctx . . . . .	154
tiledb_query_export_buffer . . . . .	154
tiledb_query_finalize . . . . .	155
tiledb_query_get_buffer_char . . . . .	155
tiledb_query_get_buffer_ptr . . . . .	156
tiledb_query_get_est_result_size . . . . .	156
tiledb_query_get_est_result_size_var . . . . .	157
tiledb_query_get_fragment_num . . . . .	157
tiledb_query_get_fragment_timestamp_range . . . . .	158
tiledb_query_get_fragment_uri . . . . .	158
tiledb_query_get_layout . . . . .	159
tiledb_query_get_range . . . . .	159
tiledb_query_get_range_num . . . . .	160
tiledb_query_get_range_var . . . . .	160
tiledb_query_import_buffer . . . . .	161
tiledb_query_result_buffer_elements . . . . .	161
tiledb_query_result_buffer_elements_vec . . . . .	162
tiledb_query_set_buffer . . . . .	163
tiledb_query_set_buffer_ptr . . . . .	163
tiledb_query_set_buffer_ptr_char . . . . .	164
tiledb_query_set_condition . . . . .	164
tiledb_query_set_layout . . . . .	165

<code>tiledb_query_set_subarray</code>	165
<code>tiledb_query_stats</code>	166
<code>tiledb_query_status</code>	166
<code>tiledb_query_submit</code>	167
<code>tiledb_query_submit_async</code>	167
<code>tiledb_query_type</code>	168
<code>tiledb_schema_get_dim_attr_status</code>	168
<code>tiledb_schema_get_enumeration_status</code>	169
<code>tiledb_schema_get_names</code>	169
<code>tiledb_schema_get_types</code>	170
<code>tiledb_schema_object</code>	170
<code>tiledb_set_context</code>	171
<code>tiledb_set_vfs</code>	171
<code>tiledb_stats_disable</code>	172
<code>tiledb_stats_dump</code>	172
<code>tiledb_stats_enable</code>	172
<code>tiledb_stats_print</code>	173
<code>tiledb_stats_raw_dump</code>	173
<code>tiledb_stats_raw_get</code>	173
<code>tiledb_stats_raw_print</code>	174
<code>tiledb_stats_reset</code>	174
<code>tiledb_subarray</code>	174
<code>tiledb_subarray-class</code>	175
<code>tiledb_subarray_to_query</code>	175
<code>tiledb_version</code>	176
<code>tiledb_vfs</code>	176
<code>tiledb_vfs-class</code>	177
<code>tiledb_vfs_close</code>	177
<code>tiledb_vfs_copy_file</code>	178
<code>tiledb_vfs_create_bucket</code>	178
<code>tiledb_vfs_create_dir</code>	179
<code>tiledb_vfs_dir_size</code>	179
<code>tiledb_vfs_empty_bucket</code>	180
<code>tiledb_vfs_file_size</code>	180
<code>tiledb_vfs_is_bucket</code>	181
<code>tiledb_vfs_is_dir</code>	181
<code>tiledb_vfs_is_empty_bucket</code>	182
<code>tiledb_vfs_is_file</code>	183
<code>tiledb_vfs_ls</code>	183
<code>tiledb_vfs_ls_recursive</code>	184
<code>tiledb_vfs_move_dir</code>	184
<code>tiledb_vfs_move_file</code>	185
<code>tiledb_vfs_open</code>	185
<code>tiledb_vfs_read</code>	186
<code>tiledb_vfs_remove_bucket</code>	186
<code>tiledb_vfs_remove_dir</code>	187
<code>tiledb_vfs_remove_file</code>	187
<code>tiledb_vfs_serialize</code>	188

tiledb_vfs_sync . . . . .	188
tiledb_vfs_touch . . . . .	189
tiledb_vfs_unserialize . . . . .	189
tiledb_vfs_write . . . . .	190
tile_order,tiledb_array_schema-method . . . . .	190
[,tiledb_array,ANY-method . . . . .	191
[,tiledb_config,ANY-method . . . . .	191
[,tiledb_filter_list,ANY-method . . . . .	192
[<-,tiledb_array,ANY,ANY,ANY-method . . . . .	193
[<-,tiledb_config,ANY,ANY,ANY-method . . . . .	194

**Index****195**


---

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)

tiledb_array_schema_get_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

tiledb_array_schema_set_allows_dups(x, value)
```

### Arguments

x	tiledb_array_schema
value	logical value

### Value

the tiledb\_array\_schema object

---

array_consolidate	<i>Consolidate fragments of a TileDB Array</i>
-------------------	--

---

### Description

This function invokes a consolidation operation. Parameters affecting the operation can be set via an optional configuration object. Start and end timestamps can also be set directly.

### Usage

```
array_consolidate(
  uri,
  cfg = NULL,
  start_time,
  end_time,
  ctx = tiledb_get_context()
)
```

**Arguments**

uri	A character value with the URI of a TileDB Array
cfg	An optional TileDB Configuration object
start_time	An optional timestamp value, if missing config default is used
end_time	An optional timestamp value, if missing config default is used
ctx	An option TileDB Context object

**Value**

NULL is returned invisibly

---

array_vacuum	<i>After consolidation, remove consolidated fragments of a TileDB Array</i>
--------------	---

---

**Description**

This function can remove fragments following a consolidation step. Note that vacuuming should *not* be run if one intends to use the TileDB *time-traveling* feature of opening arrays at particular timestamps.

**Usage**

```
array_vacuum(uri, cfg = NULL, start_time, end_time, ctx = tiledb_get_context())
```

**Arguments**

uri	A character value with the URI of a TileDB Array
cfg	An optional TileDB Configuration object
start_time	An optional timestamp value, if missing config default is used
end_time	An optional timestamp value, if missing config default is used
ctx	An option TileDB Context object

**Details**

Parameters affecting the operation can be set via an optional configuration object. Start and end timestamps can also be set directly.

**Value**

NULL is returned invisibly

---

```
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)
```

---

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; NA means no attributes will be returned.

---

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_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; the value NA_character_ signals no attributes should be returned; default is an empty character vector implying all columns are returned.

**Value**

The modified tiledb\_array object

---

```
capacity
```

*Retrieve schema capacity (for sparse fragments)*

---

**Description**

Returns the tiledb\_array schema tile capacity for sparse fragments.

**Usage**

```
capacity(object)

## S4 method for signature 'tiledb_array_schema'
capacity(object)

tiledb_array_schema_get_capacity(object)
```

**Arguments**

object	An array_schema object
--------	------------------------

**Value**

The tile capacity value

---

capacity<-                    *Sets the schema capacity (for sparse fragments)*

---

### Description

Sets the tiledb\_array schema tile capacity for sparse fragments.

### Usage

```
capacity(x) <- value

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

tiledb_array_schema_set_capacity(x, value)
```

### Arguments

x                    An array\_schema object  
value                An integer or numeric value for the new tile capacity

### Value

The modified array\_schema object

---

cell\_order,tiledb\_array\_schema-method  
*Returns the cell layout string associated with the tiledb\_array\_schema*

---

### 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	<i>Return the number of scalar values per attribute cell</i>
--------------	--

---

**Description**

Return the number of scalar values per attribute cell

**Usage**

```
cell_val_num(object)

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

tiledb_attribute_get_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)
```

---

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

---

**Description**

Return the number of scalar values per dimension cell

**Usage**

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

tiledb_dim_get_cell_val_num(object)
```

**Arguments**

object            tiledb\_dim object

**Value**

integer number of cells

---

cell\_val\_num<-            *Set the number of scalar values per attribute cell*

---

**Description**

Set the number of scalar values per attribute cell

**Usage**

```
cell_val_num(x) <- value

## S4 replacement method for signature 'tiledb_attr'
cell_val_num(x) <- value

tiledb_attribute_set_cell_val_num(x, value)
```

**Arguments**

x                    A TileDB Attribute object  
value                An integer value of number of cells

**Value**

The modified attribute is returned

---

completedBatched            *Check 'batched' query for completion*

---

**Description**

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

**Usage**

```
completedBatched(obj)
```

**Arguments**

obj                    A list object as returned by createBatched

**Value**

A logical value to indicated if the query completed

---

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"]
```

---

createBatched	<i>Create a 'batched' query object</i>
---------------	--

---

**Description**

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

**Usage**

```
createBatched(x)
```

**Arguments**

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

**Details**

The tiledb\_array object can be parameterised as usual.

**Value**

A batchedquery object, that is a list containing an external pointer to a TileDB Query object along with other support variables used by fetchBatched

---

datatype, tiledb_attr-method	<i>Return the tiledb_attr datatype</i>
------------------------------	--

---

**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 the tiledb\_domain TileDB type string*

---

**Description**

Returns the 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)
```

---

datetimes\_as\_int64      *Retrieve datetimes\_as\_int64 toggle*

---

**Description**

A tiledb\_array object may contain date and datetime objects. While their internal representation is generally shielded from the user, it can be useful to access them as the ‘native’ format which is an integer64. This function retrieves the current value of the selection variable, which has a default of FALSE.

**Usage**

```
datetimes_as_int64(object)

## S4 method for signature 'tiledb_array'
datetimes_as_int64(object)
```

**Arguments**

```
object          A tiledb_array object
```

**Value**

A logical value indicating whether datetimes\_as\_int64 is selected

---

```
datetimes_as_int64<- Set datetimes_as_int64 toggle
```

---

**Description**

A tiledb\_array object may contain date and datetime objects. While their internal representation is generally shielded from the user, it can be useful to access them as the 'native' format which is an integer64. This function sets the current value of the selection variable, which has a default of FALSE.

**Usage**

```
datetimes_as_int64(x) <- value

## S4 replacement method for signature 'tiledb_array'
datetimes_as_int64(x) <- value
```

**Arguments**

x	A tiledb_array object
value	A logical value with the selection

**Value**

The modified tiledb\_array array object

---

```
describe Describe a TileDB array schema via code to create it
```

---

**Description**

Describe a TileDB array schema via code to create it

**Usage**

```
describe(arr)
```

**Arguments**

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

**Value**

Nothing is returned as the function is invoked for the side effect of printing the schema via a sequence of R instructions to re-create it.

---

`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`*Retrieves the dimension of the tiledb\_dim domain*

---

**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

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

---

### 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,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<-	<i>Set data.frame extended return columns toggle</i>
------------	--

---

**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

---

fetchBatched	<i>Run a 'batched' query</i>
--------------	------------------------------

---

**Description**

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

**Usage**

```
fetchBatched(x, obj)
```

**Arguments**

x	A tiledb_array object
obj	A batchedquery object as returned by createBatched

**Details**

The tiledb\_array object can be parameterised as usual.

**Value**

A data.frame object with the (potentially partial) result of a batched query

---

*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 Attribute*

---

**Description**

Returns the TileDB Filter List object associated with the given TileDB Attribute

**Usage**

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

**Arguments**

object            TileDB Attribute

**Value**

a tiledb\_filter\_list object

**Examples**

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

---

*filter\_list, tiledb\_dim-method*

*Returns the TileDB Filter List object associated with the given TileDB Dimension*

---

**Description**

Returns the TileDB Filter List object associated with the given TileDB Dimension

**Usage**

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

**Arguments**

object            TileDB\_Dimension

**Value**

A TileDB\_filter\_list object

---

*filter\_list<-, tiledb\_attr-method*

*Sets the TileDB Filter List for the TileDB Attribute object*

---

**Description**

Sets the TileDB Filter List for the TileDB Attribute object

**Usage**

```
## S4 replacement method for signature 'tiledb_attr'
filter_list(x) <- value
```

**Arguments**

x	TileDB Attribute
value	TileDB Filter List

**Value**

The modified TileDB Attribute object

---

```
filter_list<-, tiledb_dim-method
```

*Sets the TileDB Filter List for the TileDB Dimension object*

---

**Description**

Sets the TileDB Filter List for the TileDB Dimension object

**Usage**

```
## S4 replacement method for signature 'tiledb_dim'
filter_list(x) <- value
```

**Arguments**

x	TileDB Dimension
value	TileDB Filter List

**Value**

The modified TileDB Dimension object

---

```
fromDataFrame
```

*Create a TileDB dense or sparse array from a given data.frame Object*

---

**Description**

The supplied data.frame object is (currently) limited to integer, numeric, or character. In addition, three datetime columns are supported with the R representations of Date, POSIXct and nanotime.

**Usage**

```

fromDataFrame(
  obj,
  uri,
  col_index = NULL,
  sparse = TRUE,
  allows_dups = sparse,
  cell_order = "COL_MAJOR",
  tile_order = "COL_MAJOR",
  filter = "ZSTD",
  capacity = 10000L,
  tile_domain = NULL,
  tile_extent = NULL,
  mode = c("ingest", "schema_only", "append"),
  filter_list = NULL,
  coords_filters = "ZSTD",
  offsets_filters = "ZSTD",
  validity_filters = "RLE",
  debug = FALSE
)

```

**Arguments**

<code>obj</code>	A data.frame object.
<code>uri</code>	A character variable with an Array URI.
<code>col_index</code>	An optional column index, either numeric with a column index, or character with a column name, designating an index column; default is NULL implying an index column is added when the array is created
<code>sparse</code>	A logical switch to select sparse (the default) or dense
<code>allows_dups</code>	A logical switch to select if duplicate values are allowed or not, default is the same value as 'sparse'.
<code>cell_order</code>	A character variable with one of the TileDB cell order values, default is "COL_MAJOR".
<code>tile_order</code>	A character variable with one of the TileDB tile order values, default is "COL_MAJOR".
<code>filter</code>	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
<code>capacity</code>	A integer value with the schema capacity, default is 10000.
<code>tile_domain</code>	An integer vector or list or NULL. If an integer vector of size two it specifies the integer domain of the row dimension; if a list then a named element is used for the dimension of the same name; or if NULL the row dimension of the obj is used.
<code>tile_extent</code>	An integer value for the tile extent of the row dimensions; if NULL the row dimension of the obj is used. Note that the <code>tile_extent</code> cannot exceed the tile domain.

mode	A character variable with possible values ‘ingest’ (for schema creation and data ingestion, the default behavior), ‘schema_only’ (to create the array schema without writing to the newly-created array) and ‘append’ (to only append to an already existing array).
filter_list	A named list specifying filter choices per column, default is an empty list object. This argument applies for all named arguments and the matching dimensions or attributes. The filter argument still applies for all unnamed arguments.
coords_filters	A character vector with filters for coordinates, default is ZSTD.
offsets_filters	A character vector with filters for coordinates, default is ZSTD.
validity_filters	A character vector with filters for coordinates, default is RLE.
debug	Logical flag to select additional output.

### Details

The created (dense or sparse) array will have as many attributes as there are columns in the `data.frame`. Each attribute will be a single column. For a sparse array, one or more columns have to be designated as dimensions.

At present, factor variables are converted to character.

### Value

Null, invisibly.

### Examples

```
uri <- tempfile()
fromDataFrame(iris, uri)
arr <- tiledb_array(uri, return_as="data.frame", extended=FALSE)
newdf <- arr[]
all.equal(iris, newdf, check.attributes=FALSE) # extra attribute on query in newdf
all.equal(as.matrix(iris), as.matrix(newdf)) # also strips attribute
```

---

fromMatrix

*Create a TileDB array from an R matrix, or return an R matrix*

---

### Description

The functions `fromMatrix` and `toMatrix` help in storing (and retrieving) matrices using a TileDB backend. In particular they help for matrices with explicit rownames.

### Usage

```
fromMatrix(obj, uri, filter = "ZSTD", capacity = 10000L)

toMatrix(uri)
```

**Arguments**

obj	A sparse matrix object.
uri	A character variable with an Array URI.
filter	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
capacity	A integer value with the schema capacity, default is 10000.

**Value**

Null, invisibly.

---

fromSparseMatrix	<i>Create (or return) a TileDB sparse array</i>
------------------	---

---

**Description**

The functions fromSparseMatrix and toSparseMatrix help in storing (and retrieving) sparse matrices using a TileDB backend.

**Usage**

```
fromSparseMatrix(
  obj,
  uri,
  cell_order = "ROW_MAJOR",
  tile_order = "ROW_MAJOR",
  filter = "ZSTD",
  capacity = 10000L
)

toSparseMatrix(uri)
```

**Arguments**

obj	A sparse matrix object.
uri	A character variable with an Array URI.
cell_order	A character variable with one of the TileDB cell order values, default is "COL_MAJOR".
tile_order	A character variable with one of the TileDB tile order values, default is "COL_MAJOR".
filter	A character variable vector, defaults to 'ZSTD', for one or more filters to be applied to each attribute;
capacity	A integer value with the schema capacity, default is 10000.

**Value**

Null, invisibly.

**Examples**

```
## Not run:
if (requireNamespace("Matrix", quietly=TRUE)) {
  library(Matrix)
  set.seed(123)      # just to fix it
  mat <- matrix(0, nrow=20, ncol=10)
  mat[sample(seq_len(200), 20)] <- seq(1, 20)
  spmat <- as(mat, "dgTMatrix") # sparse matrix in dgTMatrix format
  uri <- "sparse_matrix"
  fromSparseMatrix(spmat, uri) # now written
  chk <- toSparseMatrix(uri)   # and re-read
  print(chk)
  all.equal(spmat, chk)
}

## End(Not run)
```

---

generics

*Generic Methods*

---

**Description**

Definition of generic methods

**Usage**

```
schema(object, ...)

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

return.data.frame(x) <- value

attrs(x) <- value

raw_dump(object, ...)

domain(object, ...)

dimensions(object, ...)

attrs(object, idx, ...)

cell_order(object, ...)

tile_order(object, ...)

filter_list(object, ...)
```

```
filter_list(x) <- value
is.sparse(object, ...)
tiledb_ndim(object, ...)
name(object)
datatype(object)
config(object, ...)
tile(object)
is.integral(object)
nfilters(object)
tdb_filter(x, ...)
tdb_select(x, ...)
tdb_collect(x, ...)
```

### Arguments

object	A TileDB object
...	Currently unused
x	A TileDB Object
value	A value to be assigned
idx	An index argument

---

has_attribute	<i>Check a schema for a given attribute name</i>
---------------	--

---

### Description

Check a schema for a given attribute name

### Usage

```
has_attribute(schema, attr)
```

**Arguments**

schema	A schema for a TileDB Array
attr	A character variable with an attribute name

**Value**

A boolean value indicating if the attribute exists in the schema

---

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

---

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" environment 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.

**Details**

As this function returns a config object, its intended use is as argument to the context creating functions: `ctx <- tiledb_ctx(limitTiledBCores())`. To check that the values are set (or at a later point, still set) the config object should be retrieved via the corresponding method and this ctx object: `cfg <- config(ctx)`.

**Value**

The modified configuration object is returned invisibly.

---

max_chunk_size	<i>Returns the filter_list's max_chunk_size</i>
----------------	---

---

**Description**

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

**Usage**

```
max_chunk_size(object)

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

tiledb_filter_list_get_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)
```

---

parse\_query\_condition *Create a 'tiledb\_query\_condition' object from an expression*

---

**Description**

The grammar for query conditions is at present constraint to eight operators (" $>$ ", " $>=$ ", " $<$ ", " $<=$ ", " $=$ ", " $!=$ ", " $\%in\%$ ", " $\%nin\%$ "), and three boolean operators (" $\&\&$ ", also as " $\&$ ", (" $\|\|$ ", also as " $\|$ ", and " $!$ " for negation. Note that we locally define " $\%nin\%$ " as `Negate()` call around  $\%in\%$ ) which extends R a little for this use case.

**Usage**

```
parse_query_condition(
  expr,
  ta = NULL,
  debug = FALSE,
  strict = TRUE,
  use_int64 = FALSE
)
```

**Arguments**

expr	An expression that is understood by the TileDB grammar for query conditions.
ta	A tiledb_array object that the query condition is applied to; this argument is optional in some cases but required in some others.
debug	A boolean toogle to enable more verbose operations, defaults to 'FALSE'.
strict	A boolean toogle to, if set, errors if a non-existing attribute is selected or filtered on, defaults to 'TRUE'; if 'FALSE' a warning is shown by execution proceeds.
use_int64	A boolean toggle to switch to integer64 if integer is seen, default is false to remain as a default four-byte int

**Details**

Expressions are parsed locally by this function. The `debug=TRUE` option may help if an issue has to be diagnosed. In most cases of an erroneous parse, it generally helps to supply the `tiledb_array` providing schema information. One example are numeric and integer columns where the data type is difficult to guess. Also, when using the `"%in%"` or `"%nin%"` operators, the argument is mandatory.

**Value**

A `tiledb_query_condition` object

**Examples**

```
## Not run:
uri <- "mem://airquality" # change to on-disk for persistence
fromDataFrame(airquality, uri, col_index=c("Month", "Day")) # dense array
## query condition on dense array requires extended=FALSE
tiledb_array(uri, return_as="data.frame", extended=FALSE,
             query_condition=parse_query_condition(Temp > 90))[]

## End(Not run)
```

---

`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**

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

**Value**

The array object, invisibly

---

query_condition	<i>Retrieve query_condition value for the array</i>
-----------------	---

---

**Description**

A tiledb\_array object can have a corresponding query condition object. This methods returns it.

**Usage**

```
query_condition(object)

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

**Arguments**

object            A tiledb\_array object

**Value**

A tiledb\_query\_condition object

---

query_condition<-	<i>Set query_condition object for the array</i>
-------------------	---

---

**Description**

A tiledb\_array object can have an associated query condition object to set conditions on the read queries. This methods sets the 'query\_condition' object.

**Usage**

```
query_condition(x) <- value

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

**Arguments**

x                    A tiledb\_array object  
value                A tiledb\_query\_conditon\_object

**Value**

The modified tiledb\_array array object

---

query_layout	<i>Retrieve query_layout values for the array</i>
--------------	---

---

**Description**

A tiledb\_array object can have a corresponding query with a given layout given layout. This methods returns the selection value for 'query\_layout' as a character value.

**Usage**

```
query_layout(object)

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

**Arguments**

object            A tiledb\_array object

**Value**

A character value describing the query layout

---

query_layout<-	<i>Set query_layout return values for the array</i>
----------------	---

---

**Description**

A tiledb\_array object can have an associated query with a specific layout. This methods sets the selection value for 'query\_layout' from a character value.

**Usage**

```
query_layout(x) <- value

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

**Arguments**

x                    A tiledb\_array object  
value                A character variable for the query layout. Permitted values are "ROW\_MAJOR", "COL\_MAJOR", "GLOBAL\_ORDER", or "UNORDERD".

**Value**

The modified tiledb\_array array object

---

query_statistics	<i>Retrieve query_statistics toggle</i>
------------------	---

---

### Description

A tiledb\_array object can, if requested, return query statistics as a JSON string in an attribute 'query\_statistics' attached to the return object. The default value of the logical switch is 'FALSE'. This method returns the current value.

### Usage

```
query_statistics(object, ...)

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

### Arguments

object	A tiledb_array object
...	Currently unused

### Value

A logical value indicating whether query statistics are returned.

---

query_statistics<-	<i>Set query_statistics toggle</i>
--------------------	------------------------------------

---

### Description

A tiledb\_array object can, if requested, return query statistics as a JSON string in an attribute 'query\_statistics' attached to the return object. The default value of the logical switch is 'FALSE'. This method sets the value.

### Usage

```
query_statistics(x) <- value

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

### Arguments

x	A tiledb_array object
value	A logical value with the selection

**Value**

The modified tiledb\_array array object

---

*raw\_dump,tiledb\_array\_schema-method*

*Raw display of an array schema object*

---

**Description**

This method used the display method provided by the underlying library.

**Usage**

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

**Arguments**

object            An array\_schema object

---

*raw\_dump,tiledb\_attr-method*

*Raw display of an attribute object*

---

**Description**

This method used the display method provided by the underlying library.

**Usage**

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

**Arguments**

object            An attribute object

---

 raw\_dump,tiledb\_domain-method

*Raw display of a domain object*


---

### Description

This method used the display method provided by the underlying library.

### Usage

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

### Arguments

object	A domain object
--------	-----------------

---

 return.array

*Retrieve array return toggle*


---

### Description

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

### Usage

```
return.array(object, ...)

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

### Arguments

object	A tiledb_array object
...	Currently unused

### Value

A logical value indicating whether array return is selected

---

```
return.array<-          Set array return toggle
```

---

**Description**

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

**Usage**

```
return.array(x) <- value

## S4 replacement method for signature 'tiledb_array'
return.array(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_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_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.matrix      Retrieve matrix return toggle
```

---

**Description**

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

**Usage**

```
return.matrix(object, ...)

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

**Arguments**

object	A tiledb_array object
...	Currently unused

**Value**

A logical value indicating whether matrix return is selected

---

return.matrix<-	<i>Set matrix return toggle</i>
-----------------	---------------------------------

---

### Description

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

### Usage

```
return.matrix(x) <- value

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

### Arguments

x	A tiledb_array object
value	A logical value with the selection

### Value

The modified tiledb\_array array object

---

return_as	<i>Retrieve return_as conversion preference</i>
-----------	---

---

### Description

A tiledb\_array object can be returned as a 'list' (default), 'array', 'matrix', 'data.frame', 'data.table' or 'tibble'. This method permits to select a preference for the returned object. The default value of 'asis' means that no conversion is performed.

### Usage

```
return_as(object, ...)
```

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

### Arguments

object	A tiledb_array object
...	Currently unused

**Value**

A character value indicating the preferred conversion where the value is one of 'asis' (the default), 'array', 'matrix', 'data.frame', 'data.table', or 'tibble'.

---

return_as<-	<i>Retrieve return_as conversion preference</i>
-------------	---

---

**Description**

A tiledb\_array object can be returned as a 'list' (default), 'array', 'matrix', 'data.frame', 'data.table' or 'tibble'. This method This methods permits to set a preference of returning a list, array, matrix, data.frame, a data.table, or a tibble. The default value of "asis" means that no conversion is performed and a list is returned.

**Usage**

```
return_as(x) <- value

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

**Arguments**

x	A tiledb_array object
value	A character value with the selection

**Value**

The modified tiledb\_array array object

---

r_to_tiledb_type	<i>Look up TileDB type corresponding to the type of an R object</i>
------------------	---

---

**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

---

```
save_allocation_size_preference
      Store allocation size preference
```

---

**Description**

Save (or load) allocation size default preference in an optional config file

**Usage**

```
save_allocation_size_preference(value)

load_allocation_size_preference()

get_allocation_size_preference()

set_allocation_size_preference(value)
```

**Arguments**

value            A numeric value with the desired allocation size (in bytes).

**Details**

When retrieving data from sparse arrays, allocation sizes cannot be determined *ex ante* as the degree of sparsity is unknown. A configuration value can aid in providing an allocation size value. These functions let the user store such a value for retrieval by their package or script code. The preference will be encoded in a configuration file as R (version 4.0.0 or later) allows a user- and package specific configuration files. These helper functions sets and retrieve the value, respectively, or retrieve the cached value from the package environment where is it set at package load.

The value will be stored as a character value and reparsed so ‘1e6’ and ‘1000000’ are equivalent, and the fixed (but adjustable) number of digits for numerical precision *use for formatting* will impact the writing. This should have no effect on standard allocation sizes.

The value is used as a limit *per column* so total memory use per query will a multiple of this value, and increasing in dimension and attribute count.

A fallback value of 10 mb is used if no user value is set.

**Value**

For the setter, TRUE is returned invisibly but the function is invoked for the side effect of storing the value. For the getters, the value as a numeric.

**Note**

This function requires R version 4.0.0 or later to utilise the per-user config directory accessor function. For older R versions, a fallback from the TileDB configuration object is used.

---

save\_return\_as\_preference  
*Store object conversion preference*

---

### Description

Save (or load) 'return\_as' conversion preference in an optional config file

### Usage

```
save_return_as_preference(  
  value = c("asis", "array", "matrix", "data.frame", "data.table", "tibble")  
)  
  
load_return_as_preference()  
  
get_return_as_preference()  
  
set_return_as_preference(  
  value = c("asis", "array", "matrix", "data.frame", "data.table", "tibble")  
)
```

### Arguments

value            A character variable with one of the six permitted values

### Details

The tiledb\_array object can set a preference for conversion for each retrieved object. This preference can also be encoded in a configuration file as R (version 4.0.0 or later) allows a user- and package specific configuration files. These helper functions sets and retrieve the value, respectively, or retrieve the cached value from the package environment where is it set at package load.

Note that the value must be one of 'asis' (the default), 'array', 'matrix', 'data.frame', 'data.table' or 'tibble'. The latter two require the corresponding package to be installed.

### Value

For the setter, TRUE is returned invisibly but the function is invoked for the side effect of storing the value. For either getter, the character value.

### Note

This function requires R version 4.0.0 or later to utilise the per-user config directory accessor function. For older R versions, please set the attribute directly when creating the tiledb\_array object, or via the return\_as() method.

---

schema,character-method

*Return a schema from a URI character value*

---

### Description

Return a schema from a URI character value

### Usage

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

### Arguments

object	A character variable with a URI
...	Extra parameters such as 'enckey', the encryption key

### Value

The scheme for the object

---

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_check	<i>Check the schema for correctness</i>
--------------	---

---

**Description**

Returns the tiledb\_array schema for correctness

**Usage**

```

schema_check(object)

## S4 method for signature 'tiledb_array_schema'
schema_check(object)

check(object)

## S4 method for signature 'tiledb_array_schema'
check(object)

tiledb_array_schema_check(object)

```

**Arguments**

object            An array\_schema object

**Value**

The boolean value TRUE is returned for a correct schema; for an incorrect schema an error condition is triggered.

---

selected_points	<i>Retrieve selected_points values for the array</i>
-----------------	--

---

**Description**

A tiledb\_array object can have a range selection for each dimension attribute. This methods returns the selection value for 'selected\_points' and returns a list (with one element per dimension) of vectors where each row describes one selected points. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

**Usage**

```

selected_points(object)

## S4 method for signature 'tiledb_array'
selected_points(object)

```

**Arguments**

object            A tiledb\_array object

**Value**

A list which can contain a vector for each dimension

---

selected\_points<-        *Set selected\_points 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\_points' which is a list (with one element per dimension) of two-column matrices where each row describes one pair of minimum and maximum values. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

**Usage**

```
selected_points(x) <- value

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

**Arguments**

x                    A tiledb\_array object

value                A list of vectors where each list element 'i' corresponds to the dimension attribute 'i'.

**Value**

The modified tiledb\_array array 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. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

**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. Alternatively, the list can be named with the names providing the match to the corresponding dimension.

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

---

**Description**

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

**Usage**

```
set_max_chunk_size(object, value)

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

tiledb_filter_list_set_max_chunk_size(object, value)
```

**Arguments**

object	tiledb_filter_list
value	A numeric value

**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\_dim-method

*Prints a dimension object*

---

### Description

Prints a dimension object

### Usage

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

### Arguments

object            A dimension object

---

show,tiledb\_domain-method  
*Prints a domain object*

---

### **Description**

Prints a domain object

### **Usage**

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

### **Arguments**

object            A domain object

---

show,tiledb\_filter-method  
*Prints a filter object*

---

### **Description**

Prints a filter object

### **Usage**

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

### **Arguments**

object            A filter object

---

show,tiledb\_filter\_list-method  
*Prints a filter\_list object*

---

**Description**

Prints a filter\_list object

**Usage**

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

**Arguments**

object            A filter\_list object

---

show,tiledb\_group-method  
*Display the TileDB Group object to STDOUT*

---

**Description**

Display the TileDB Group object to STDOUT

**Usage**

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

**Arguments**

object            tiledb\_group object

---

statusBatched	<i>Return 'batched' status</i>
---------------	--------------------------------

---

**Description**

Batched queries return an initial result set even when it is incomplete. Where the normal retrieval process will loop in place to complete a (potentially large) result set, this function will return a result (which may be part of a larger result set) allowing the user to assemble all part.

**Usage**

```
statusBatched(obj)
```

**Arguments**

obj	A list object as returned by createBatched
-----	--

**Value**

The Query status as a character variable

---

strings_as_factors	<i>Retrieve strings_as_factors conversion toggle</i>
--------------------	--

---

**Description**

A tiledb\_array object containing character column can have those converted to factors variables. This methods returns the selection value for 'strings\_as\_factors'.

**Usage**

```
strings_as_factors(object)

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

**Arguments**

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

**Value**

A logical value indicating whether an strings\_as\_factors return is selected

---

```
strings_as_factors<- Set strings_as_factors return toggle
```

---

### Description

A tiledb\_array object containing character column can have those converted to factors variables. This methods sets the selection value for 'strings\_as\_factors'.

### Usage

```
strings_as_factors(x) <- value

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

### Arguments

x	A tiledb_array object
value	A logical value with the selection

### Value

The modified tiledb\_array array object

---

```
tdb_collect, tiledb_array-method
Collect the query results to finalize piped expression
```

---

### Description

Collect the query results to finalize piped expression

### Usage

```
## S4 method for signature 'tiledb_array'
tdb_collect(x, ...)
```

### Arguments

x	A tiledb_array object as first argument, permitting piping
...	Ignored

### Value

The object returning from a tiledb\_array query (the type of which can be set via the return preference mechanism, see the help for "[" accessor)

---



*Filter from array for query via logical conditions*

---

**Description**

Filter from array for query via logical conditions

**Usage**

```
## S4 method for signature 'tiledb_array'
tdb_filter(x, ..., strict = TRUE)
```

**Arguments**

x	A tiledb_array object as first argument, permitting piping
...	One or more expressions that are parsed as query_condition objects
strict	A boolean toggle to, if set, errors if a non-existing attribute is selected or filtered on, defaults to 'TRUE'; if 'FALSE' a warning is shown by execution proceeds.

**Value**

The tiledb\_array object, permitting piping

---

*Select attributes from array for query*

---

**Description**

Select attributes from array for query

**Usage**

```
## S4 method for signature 'tiledb_array'
tdb_select(x, ...)
```

**Arguments**

x	A tiledb_array object as first argument, permitting piping
...	One or more attributes of the query

**Value**

The tiledb\_array object, permitting piping

---

```
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            Constructs a tiledb_array object backed by a persisted tiledb array uri
```

---

**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(),
  selected_points = list(),
```

```

query_layout = character(),
datetimes_as_int64 = FALSE,
encryption_key = character(),
as.matrix = FALSE,
as.array = FALSE,
query_condition = new("tiledb_query_condition"),
timestamp_start = as.POSIXct(double(), origin = "1970-01-01"),
timestamp_end = as.POSIXct(double(), origin = "1970-01-01"),
return_as = get_return_as_preference(),
query_statistics = FALSE,
strings_as_factors = getOption("stringsAsFactors", FALSE),
keep_open = FALSE,
sil = list(),
dumpbuffers = character(),
buffers = list(),
ctx = tiledb_get_context()
)

tiledb_dense(...)

tiledb_sparse(...)

```

### 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, the special value NA_character_ has the opposite effect and implies no attributes are returned.
extended	optional logical switch selecting wide 'data.frame' format, defaults to TRUE
selected_ranges	optional A list with matrices where each matrix i describes the (min,max) pair of ranges selected for dimension i
selected_points	optional A list with vectors where each vector i describes the points selected in dimension i
query_layout	optional A value for the TileDB query layout, defaults to an empty character variable indicating no special layout is set
datetimes_as_int64	optional A logical value selecting date and datetime value representation as 'raw' integer64 and not as Date, POSIXct or nanotime objects.
encryption_key	optional A character value with an AES-256 encryption key in case the array was written with encryption.

<code>as.matrix</code>	optional logical switch, defaults to "FALSE"; currently limited to dense matrices; in the case of multiple attributes in query a list of matrices is returned
<code>as.array</code>	optional logical switch, defaults to "FALSE"; in the case of multiple attributes in query a list of arrays is returned
<code>query_condition</code>	optional <code>tiledb_query_condition</code> object, by default uninitialized without a condition; this functionality requires TileDB 2.3.0 or later
<code>timestamp_start</code>	optional A POSIXct Datetime value determining the inclusive time point at which the array is to be opened. No fragments written earlier will be considered.
<code>timestamp_end</code>	optional A POSIXct Datetime value determining the inclusive time point until which the array is to be opened. No fragments written earlier later be considered.
<code>return_as</code>	optional A character value with the desired <code>tiledb_array</code> conversion, permitted values are 'asis' (default, returning a list of columns), 'array', 'matrix', 'data.frame', 'data.table', 'tibble', 'arrow_table', or 'arrow' (as an alias for 'arrow_table'; here 'data.table', 'tibble' and 'arrow' require the respective packages to be installed. The existing <code>as.*</code> arguments take precedent over this.
<code>query_statistics</code>	optional A logical value, defaults to 'FALSE'; if 'TRUE' the query statistics are returned (as a JSON string) via the attribute 'query_statistics' of the return object.
<code>strings_as_factors</code>	An optional logical to convert character columns to factor type; defaults to the value of <code>getOption("stringsAsFactors", FALSE)</code> .
<code>keep_open</code>	An optional logical to not close after read or write
<code>sil</code>	optional A list, by default empty to store schema information when query objects are parsed.
<code>dumpbuffers</code>	An optional character variable with a directory name (relative to <code>/dev/shm</code> ) for writing out results buffers (for internal use / testing)
<code>buffers</code>	An optional list with full pathnames of shared memory buffers to read data from
<code>ctx</code>	optional <code>tiledb_ctx</code>
<code>...</code>	Used as a pass-through for <code>tiledb_dense</code> and <code>tiledb_sparse</code> aliasing

## Value

`tiledb_array` object

---

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

---

### Description

This class replaces the earlier (and now removed) `tiledb_dense` and `tiledb_sparse` and provides equivalent functionality based on a refactored implementation utilising newer TileDB features.

### Slots

`ctx` A TileDB context object

`uri` A character description with the array URI

`is.sparse` A logical value whether the array is sparse or not

`as.data.frame` A logical value

`attrs` A character vector to select particular column ‘attributes’; default is an empty character vector implying ‘all’ columns, the special value `NA_character_` has the opposite effect and selects ‘none’.

`extended` A logical value, defaults to TRUE, indicating whether index columns are returned as well.

`selected_ranges` An optional list with matrices where each matrix `i` describes the (min,max) pair of ranges for dimension `i`

`selected_points` An optional list with vectors where each vector `i` describes the selected points for dimension `i`

`query_layout` An optional character value

`datetimes_as_int64` A logical value

`encryption_key` A character value

`as.matrix` A logical value

`as.array` A logical value

`query_condition` A Query Condition object

`timestamp_start` A POSIXct datetime variable for the inclusive interval start

`timestamp_end` A POSIXct datetime variable for the inclusive interval start

`return_as` A character value with the desired `tiledb_array` conversion, permitted values are ‘asis’ (default, returning a list of columns), ‘array’, ‘matrix’, ‘data.frame’, ‘data.table’ ‘tibble’, ‘arrow\_table’ or ‘arrow’ (where the last two are synonyms); note that ‘data.table’, ‘tibble’ and ‘arrow’ require the respective packages to be installed.

`query_statistics` A logical value, defaults to ‘FALSE’; if ‘TRUE’ the query statistics are returned (as a JSON string) via the attribute ‘query\_statistics’ of the return object.

`sil` An optional and internal list object with schema information, used for parsing queries.

`dumpbuffers` An optional character variable with a directory name (relative to `/dev/shm`) for writing out results buffers (for internal use / testing)

`buffers` An optional list with full pathnames of shared memory buffers to read data from

`strings_as_factors` An optional logical to convert character columns to factor type

`keep_open` An optional logical to not close after read or write

`ptr` External pointer to the underlying implementation

---

tiledb\_array\_apply\_aggregate

*Run an aggregate query on the given (sparse) array and attribute*

---

### Description

For dense arrays, use tiledb\_query\_apply\_aggregate after setting an appropriate subarray.

### Usage

```
tiledb_array_apply_aggregate(
  array,
  attrname,
  operation = c("Count", "NullCount", "Min", "Max", "Mean", "Sum"),
  nullable = TRUE
)
```

### Arguments

array	A TileDB Array object
attrname	The name of an attribute
operation	The name of aggregation operation
nullable	A boolean toggle whether the attribute is nullable

### Value

The value of the aggregation

---

tiledb\_array\_close      *Close a TileDB Array*

---

### Description

Close a TileDB Array

### Usage

```
tiledb_array_close(arr)
```

### Arguments

arr	A TileDB Array object as for example returned by tiledb_array()
-----	---

### Value

The TileDB Array object but closed

---

tiledb\_array\_create     *Creates a new TileDB array given an input schema.*

---

### Description

Creates a new TileDB array given an input schema.

### Usage

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

### Arguments

uri	URI specifying path to create the TileDB array object
schema	tiledb_array_schema object
encryption_key	optional A character value with an AES-256 encryption key in case the array should be encryption.

### 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\_delete\_fragments     *Delete fragments written between the start and end times given*

---

### Description

Delete fragments written between the start and end times given

### Usage

```
tiledb_array_delete_fragments(  
  arr,  
  ts_start,  
  ts_end,  
  ctx = tiledb_get_context()  
)
```

**Arguments**

<code>arr</code>	A TileDB Array object as for example returned by <code>tiledb_array()</code>
<code>ts_start</code>	A Datetime object that will be converted to millisecond granularity
<code>ts_end</code>	A Datetime object that will be converted to millisecond granularity
<code>ctx</code>	A <code>tiledb_ctx</code> object (optional)

**Value**

A boolean indicating success

---

`tiledb_array_get_non_empty_domain_from_index`

*Get the non-empty domain from a TileDB Array by index*

---

**Description**

This functions works for both fixed- and variable-sized dimensions and switches internally.

**Usage**

```
tiledb_array_get_non_empty_domain_from_index(arr, idx)
```

**Arguments**

<code>arr</code>	A TileDB Array
<code>idx</code>	An integer index between one the number of dimensions

**Value**

A two-element object is returned describing the domain of selected dimension; it will either be a numeric vector in case of a fixed-size fixed-sized dimensions, or a character vector for a variable-sized one.

---

`tiledb_array_get_non_empty_domain_from_name`*Get the non-empty domain from a TileDB Array by name*

---

**Description**

This functions works for both fixed- and variable-sized dimensions and switches internally.

**Usage**

```
tiledb_array_get_non_empty_domain_from_name(arr, name)
```

**Arguments**

<code>arr</code>	A TileDB Array
<code>name</code>	An character variable with a dimension name

**Value**

A two-element object is returned describing the domain of selected dimension; it will either be a numeric vector in case of a fixed-size fixed-sized dimensions, or a characer vector for a variable-sized one.

---

`tiledb_array_has_enumeration`*Check for Enumeration (aka Factor aka Dictionary)*

---

**Description**

Check for Enumeration (aka Factor aka Dictionary)

**Usage**

```
tiledb_array_has_enumeration(arr)
```

**Arguments**

<code>arr</code>	A TileDB Array object
------------------	-----------------------

**Value**

A boolean indicating if the array has homogeneous domains

---

tiledb\_array\_is\_heterogeneous

*Check for Heterogeneous Domain*

---

**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\_is\_open    *Test if TileDB Array is open*

---

**Description**

Test if TileDB Array is open

**Usage**

```
tiledb_array_is_open(arr)
```

**Arguments**

arr                    A TileDB Array object as for example returned by tiledb\_array()

**Value**

A boolean indicating whether the TileDB Array object is open

---

tiledb\_array\_open        *Open a TileDB Array*

---

**Description**

Open a TileDB Array

**Usage**

```
tiledb_array_open(
  arr,
  type = if (tiledb_version(TRUE) >= "2.12.0") c("READ", "WRITE", "DELETE",
    "MODIFY_EXCLUSIVE") else c("READ", "WRITE")
)
```

**Arguments**

arr                    A TileDB Array object as for example returned by tiledb\_array()  
 type                   A character value that must be either 'READ', 'WRITE' or (for TileDB 2.12.0 or later) 'DELETE' or 'MODIFY\_EXCLUSIVE'

**Value**

The TileDB Array object but opened for reading or writing

---

tiledb\_array\_open\_at *Open a TileDB Array at Timestamp*

---

### Description

Open a TileDB Array at Timestamp

### Usage

```
tiledb_array_open_at(arr, type = c("READ", "WRITE"), timestamp)
```

### Arguments

arr	A TileDB Array object as for example returned by tiledb_array()
type	A character value that must be either 'READ' or 'WRITE'
timestamp	A Datetime object that will be converted to millisecond granularity

### Value

The TileDB Array object but opened for reading or writing

---

tiledb\_array\_schema *Constructs a tiledb\_array\_schema object*

---

### 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,
  validity_filter_list = NULL,
  capacity = 10000L,
  allows_dups = FALSE,
  enumerations = 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)  
 validity\_filter\_list  
     (optional)  
 capacity (optional)  
 allows\_dups (optional, requires 'sparse' to be TRUE)  
 enumerations (optional) named list of enumerations  
 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

*An S4 class for the TileDB array schema*

---

**Description**

An S4 class for the TileDB array schema

**Slots**

ptr An external pointer to the underlying implementation  
 arrptr An optional external pointer to the underlying array, or NULL if missing

---

```
tiledb_array_schema_evolution
```

*Creates a 'tiledb\_array\_schema\_evolution' object*

---

### Description

Creates a 'tiledb\_array\_schema\_evolution' object

### Usage

```
tiledb_array_schema_evolution(ctx = tiledb_get_context())
```

### Arguments

ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved
-----	---

### Value

A 'array\_schema\_evolution' object

---

```
tiledb_array_schema_evolution-class
```

*An S4 class for a TileDB ArraySchemaEvolution object*

---

### Description

An S4 class for a TileDB ArraySchemaEvolution object

### Slots

ptr An external pointer to the underlying implementation

---

 tiledb\_array\_schema\_evolution\_add\_attribute

*Add an Attribute to a TileDB Array Schema Evolution object*


---

**Description**

Add an Attribute to a TileDB Array Schema Evolution object

**Usage**

```
tiledb_array_schema_evolution_add_attribute(object, attr)
```

**Arguments**

object	A TileDB 'array_schema_evolution' object
attr	A TileDB attribute

**Value**

The modified 'array\_schema\_evolution' object, invisibly

---

tiledb\_array\_schema\_evolution\_add\_enumeration

*Add an Enumeration to a TileDB Array Schema Evolution object*


---

**Description**

Add an Enumeration to a TileDB Array Schema Evolution object

**Usage**

```
tiledb_array_schema_evolution_add_enumeration(
  object,
  name,
  enums,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

**Arguments**

object	A TileDB 'array_schema_evolution' object
name	A character value with the name for the Enumeration
enums	A character vector
ordered	(optional) A boolean switch whether the enumeration is ordered
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

The modified 'array\_schema\_evolution' object, invisibly

---

tiledb\_array\_schema\_evolution\_add\_enumeration\_empty  
*Evolve an Array Schema by adding an empty Enumeration*

---

**Description**

Evolve an Array Schema by adding an empty Enumeration

**Usage**

```
tiledb_array_schema_evolution_add_enumeration_empty(
  ase,
  enum_name,
  type_str = "ASCII",
  cell_val_num = NA_integer_,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

**Arguments**

ase	An ArraySchemaEvolution object
enum_name	A character value with the Enumeration name
type_str	A character value with the TileDB type, defaults to 'ASCII'
cell_val_num	An integer with number values per cell, defaults to NA_integer_ to flag the NA value use for character values
ordered	A logical value indicating standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

---

tiledb\_array\_schema\_evolution\_array\_evolve  
*Evolve an Array Schema*

---

**Description**

Evolve an Array Schema

**Usage**

```
tiledb_array_schema_evolution_array_evolve(object, uri)
```

**Arguments**

object            A TileDB 'array\_schema\_evolution' object  
 uri              A character variable with an URI

**Value**

The modified 'array\_schema\_evolution' object, invisibly

tiledb\_array\_schema\_evolution\_drop\_attribute

*Drop an attribute given by name from a TileDB Array Schema Evolution object*

**Description**

Drop an attribute given by name from a TileDB Array Schema Evolution object

**Usage**

tiledb\_array\_schema\_evolution\_drop\_attribute(object, attrname)

**Arguments**

object            A TileDB 'array\_schema\_evolution' object  
 attrname         A character variable with an attribute name

**Value**

The modified 'array\_schema\_evolution' object, invisibly

tiledb\_array\_schema\_evolution\_drop\_enumeration

*Drop an Enumeration given by name from a TileDB Array Schema Evolution object*

**Description**

Drop an Enumeration given by name from a TileDB Array Schema Evolution object

**Usage**

tiledb\_array\_schema\_evolution\_drop\_enumeration(object, attrname)

**Arguments**

object	A TileDB 'array_schema_evolution' object
attrname	A character variable with an attribute name

**Value**

The modified 'array\_schema\_evolution' object, invisibly

---

tiledb\_array\_schema\_evolution\_extend\_enumeration

*Extend an Evolution via Array Schema Evolution*

---

**Description**

Extend an Evolution via Array Schema Evolution

**Usage**

```
tiledb_array_schema_evolution_extend_enumeration(
  ase,
  array,
  enum_name,
  new_values,
  nullable = FALSE,
  ordered = FALSE,
  ctx = tiledb_get_context()
)
```

**Arguments**

ase	An ArraySchemaEvolution object
array	A TileDB Array object
enum_name	A character value with the Enumeration name
new_values	A character vector with the new Enumeration values
nullable	A logical value indicating if the Enumeration can contain missing values (with a default of FALSE)
ordered	A logical value indicating standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

---

`tiledb_array_schema_set_coords_filter_list`*Set a Filter List for Coordinate of a TileDB Schema*

---

**Description**

Set a Filter List for Coordinate of a TileDB Schema

**Usage**

```
tiledb_array_schema_set_coords_filter_list(sch, fl)
```

**Arguments**

<code>sch</code>	A TileDB Array Schema object
<code>fl</code>	A TileDB Filter List object

**Value**

The modified Array Schema object

---

`tiledb_array_schema_set_enumeration_empty`*Add an empty Enumeration to a Schema*

---

**Description**

Add an empty Enumeration to a Schema

**Usage**

```
tiledb_array_schema_set_enumeration_empty(  
  schema,  
  attr,  
  enum_name,  
  type_str = "ASCII",  
  cell_val_num = NA_integer_,  
  ordered = FALSE,  
  ctx = tiledb_get_context()  
)
```

**Arguments**

schema	An Array Schema
attr	An Attribute for which an empty Enumeration will be added
enum_name	A character value with the Enumeration name
type_str	A character value with the TileDB type, defaults to 'ASCII'
cell_val_num	An integer with number values per cell, defaults to NA_integer_ to flag the NA value use for character values
ordered	A logical value indicated standard factor (when FALSE, the default) or ordered (when TRUE)
ctx	Optional tiledb_ctx object

---

tiledb\_array\_schema\_set\_offsets\_filter\_list

*Set a Filter List for Variable-Sized Offsets of a TileDB Schema*

---

**Description**

Set a Filter List for Variable-Sized Offsets of a TileDB Schema

**Usage**

```
tiledb_array_schema_set_offsets_filter_list(sch, fl)
```

**Arguments**

sch	A TileDB Array Schema object
fl	A TileDB Filter List object

**Value**

The modified Array Schema object

---

tiledb\_array\_schema\_set\_validity\_filter\_list  
*Set a Filter List for Validity of a TileDB Schema*

---

**Description**

Set a Filter List for Validity of a TileDB Schema

**Usage**

```
tiledb_array_schema_set_validity_filter_list(sch, fl)
```

**Arguments**

sch	A TileDB Array Schema object
fl	A TileDB Filter List object

**Value**

The modified Array Schema object

---

tiledb\_array\_schema\_version  
*Check the version of the array schema*

---

**Description**

Returns the (internal) version of the tiledb\_array schema

**Usage**

```
tiledb_array_schema_version(object)
```

**Arguments**

object	An array_schema object
--------	------------------------

**Value**

An integer value describing the internal schema format version

---

tiledb\_array\_upgrade\_version

*Upgrade an Array to the current TileDB Array Schema Format*


---

### Description

Upgrade an Array to the current TileDB Array Schema Format

### Usage

```
tiledb_array_upgrade_version(array, config = NULL, ctx = tiledb_get_context())
```

### Arguments

array	A TileDB Array object
config	A TileDB Configuration (optional, default NULL)
ctx	A tiledb_ctx object (optional)

### Value

Nothing is returned as the function is invoked for its side effect

---

tiledb\_arrow\_array\_ptr

*(Deprecated) Allocate (or Release) Arrow Array and Schema Pointers*


---

### Description

These functions allocate (and free) appropriate pointer objects for, respectively, Arrow array and schema objects. These functions are deprecated and will be removed, it is recommended to rely directly on the nanoarrow replacements.

### Usage

```
tiledb_arrow_array_ptr()
tiledb_arrow_schema_ptr()
tiledb_arrow_array_del(ptr)
tiledb_arrow_schema_del(ptr)
```

### Arguments

ptr	A external pointer object previously allocated with these functions
-----	---

**Value**

The allocating functions return the requested pointer

---

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,
  nullable = FALSE,
  enumeration = NULL,
  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")) An optional tiledb_filter_list object
ncells	(default 1) The number of cells, use NA to signal variable length
nullable	(default FALSE) A logical switch whether the attribute can have missing values
enumeration	(default NULL) A character vector of dictionary values
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      *An S4 class for a TileDB attribute*

---

**Description**

An S4 class for a TileDB attribute

**Slots**

ptr External pointer to the underlying implementation

---

tiledb\_attribute\_get\_cell\_size  
*Get the TileDB Attribute cell size*

---

**Description**

Get the TileDB Attribute cell size

**Usage**

tiledb\_attribute\_get\_cell\_size(attr)

**Arguments**

attr                    A TileDB Attribute object

**Value**

A numeric value with the cell size

---

tiledb\_attribute\_get\_enumeration  
*Get the TileDB Attribute Enumeration*

---

**Description**

Get the TileDB Attribute Enumeration

**Usage**

tiledb\_attribute\_get\_enumeration(attr, arr, ctx = tiledb\_get\_context())

tiledb\_attribute\_get\_enumeration\_ptr(attr, arrptr, ctx = tiledb\_get\_context())

**Arguments**

<code>attr</code>	A TileDB Attribute object
<code>arr</code>	A Tiledb Array object
<code>ctx</code>	A Tiledb Context object (optional)
<code>arrptr</code>	A Tiledb Array object pointer

**Value**

A character vector with the enumeration (of length zero if none)

`tiledb_attribute_get_fill_value`

*Get the fill value for a TileDB Attribute*

**Description**

Get the fill value for a TileDB Attribute

**Usage**

`tiledb_attribute_get_fill_value(attr)`

**Arguments**

<code>attr</code>	A TileDB Attribute object
-------------------	---------------------------

**Value**

The fill value for the attribute

`tiledb_attribute_get_nullable`

*Get the TileDB Attribute Nullable flag value*

**Description**

Get the TileDB Attribute Nullable flag value

**Usage**

`tiledb_attribute_get_nullable(attr)`

**Arguments**

<code>attr</code>	A TileDB Attribute object
-------------------	---------------------------

**Value**

A boolean value with the 'Nullable' status

---

tiledb\_attribute\_has\_enumeration

*Test if TileDB Attribute has an Enumeration*

---

**Description**

Test if TileDB Attribute has an Enumeration

**Usage**

```
tiledb_attribute_has_enumeration(attr, ctx = tiledb_get_context())
```

**Arguments**

attr	A TileDB Attribute object
ctx	A Tiledb Context object (optional)

**Value**

A logical value indicating if the attribute has an enumeration

---

tiledb\_attribute\_is\_ordered\_enumeration\_ptr

*Check if TileDB Attribute Enumeration is Ordered*

---

**Description**

Check if TileDB Attribute Enumeration is Ordered

**Usage**

```
tiledb_attribute_is_ordered_enumeration_ptr(
  attr,
  arrptr,
  ctx = tiledb_get_context()
)
```

**Arguments**

attr	A Tiledb Array object
arrptr	A Tiledb Array object pointer
ctx	A Tiledb Context object (optional)

**Value**

A character vector with the enumeration (of length zero if none)

tiledb\_attribute\_is\_variable\_sized

*Check whether TileDB Attribute is variable-sized*

**Description**

Check whether TileDB Attribute is variable-sized

**Usage**

```
tiledb_attribute_is_variable_sized(attr)
```

**Arguments**

attr            A TileDB Attribute object

**Value**

A boolean value indicating variable-size or not

tiledb\_attribute\_set\_enumeration\_name

*Set a TileDB Attribute Enumeration Name*

**Description**

Set a TileDB Attribute Enumeration Name

**Usage**

```
tiledb_attribute_set_enumeration_name(
  attr,
  enum_name,
  ctx = tiledb_get_context()
)
```

**Arguments**

attr            A TileDB Attribute object  
 enum\_name      A character value with the enumeration value  
 ctx            A Tiledb Context object (optional)

**Value**

The modified TileDB Attribute object

tiledb\_attribute\_set\_fill\_value

*Set the fill value for a TileDB Attribute*

---

**Description**

Set the fill value for a TileDB Attribute

**Usage**

```
tiledb_attribute_set_fill_value(attr, value)
```

**Arguments**

attr	A TileDB Attribute object
value	A fill value

**Value**

NULL is returned invisibly

---

tiledb\_attribute\_set\_nullable

*Set the TileDB Attribute Nullable flags*

---

**Description**

Set the TileDB Attribute Nullable flags

**Usage**

```
tiledb_attribute_set_nullable(attr, flag)
```

**Arguments**

attr	A TileDB Attribute object
flag	A boolean flag to turn 'Nullable' on or off

**Value**

Nothing is returned

---

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

---

### Description

Note that for actually setting persistent values, the (altered) config object needs to be used to create (or update) the tiledb\_ctx object. Similarly, to check whether values are set, one should use the config method of the tiledb\_ctx object. Examples for this are `ctx <- tiledb_ctx(limitTileDBCores())` to use updated configuration values to create a context object, and `cfg <- config(ctx)` to retrieve it.

### 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\_as\_built\_json

*Return the 'AsBuilt' JSON string*

---

### **Description**

Return the 'AsBuilt' JSON string

### **Usage**

```
tiledb_config_as_built_json()
```

### **Value**

The JSON string containing 'AsBuilt' information

### **Examples**

```
if (tiledb_version(TRUE) > "2.17")
  txt <- tiledb::tiledb_config_as_built_json()
## now eg either one of
##   sapply(jsonlite::fromJSON(txt)$as_built$parameters$storage_backends, \(x) x[[1]])
##   sapply(RcppSimdJson::fparse(txt)$as_built$parameters$storage_backends, \(x) x[[1]])
## will return a named vector such as
##   c(azure = FALSE, gcs = FALSE, hdfs = FALSE, s3 = TRUE)
```

---

tiledb\_config\_as\_built\_show

*Display the 'AsBuilt' JSON string*

---

### **Description**

Display the 'AsBuilt' JSON string

### **Usage**

```
tiledb_config_as_built_show()
```

### **Value**

Nothing is returned but as a side-effect the 'AsBuilt' string is displayed

---

tiledb\_config\_load      *Load a saved tiledb\_config file from disk*

---

**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      *Save a tiledb\_config object to a local text file*

---

**Description**

Save a tiledb\_config object to 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\_config\_unset     *Unset a TileDB Config parameter to its default value*

---

**Description**

Unset a TileDB Config parameter to its default value

**Usage**

```
tiledb_config_unset(config, param)
```

**Arguments**

config	A TileDB Config object
param	A character variable with the parameter name

**Value**

The modified TileDB Config object

---

tiledb\_ctx             *Creates a tiledb\_ctx object*

---

**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      *Sets a string:string "tag" on the Ctx*

---

**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\_ctx\_stats      *Return context statistics as a JSON string*

---

**Description**

Return context statistics as a JSON string

**Usage**

```
tiledb_ctx_stats(object = tiledb_get_context())
```

**Arguments**

object	A tiledb_ctx object
--------	---------------------

**Value**

A JSON-formatted string with context statistics

---

`tiledb_datatype_R_type`*Map from TileDB type to R datatype*

---

**Description**

This function maps from the TileDB types to the (fewer) key datatypes in R. This can be lossy as TileDB integers range from (signed and unsigned) 8 to 64 bit whereas R only has (signed) 32 bit values. Similarly, R only has 64 bit doubles whereas TileDB has 32 and 64 bit floating point types. TileDB also has more character encodings, and the full range of (NumPy) date and time types.

**Usage**

```
tiledb_datatype_R_type(datatype)
```

**Arguments**

`datatype`      A string describing one TileDB datatype

**Value**

A string describing the closest match for an R datatype

---

`tiledb_delete_metadata`*Delete a TileDB Array Metadata object given by key*

---

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

---

**Description**

Constructs a tiledb\_dim object

**Usage**

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

**Arguments**

name	The dimension name / label string. This argument is required.
domain	The dimension (inclusive) domain. The domain of a dimension is defined by a (lower bound, upper bound) vector. For type ASCII, NULL is expected.
tile	The tile dimension tile extent. For type ASCII, NULL is expected.
type	The dimension TileDB datatype string.
filter_list	An optional tiledb_filter_list object, default is no filter
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      *An S4 class for a TileDB dimension object*

**Description**

An S4 class for a TileDB dimension object

**Slots**

ptr An external pointer to the underlying implementation

tiledb\_domain      *Constructs a tiledb\_domain object*

**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      *An S4 class for a TileDB domain*

**Description**

An S4 class for a TileDB domain

**Slots**

ptr External pointer to the underlying implementation

---

`tiledb_domain_get_dimension_from_index`*Returns a Dimension indicated by index for the given TileDB Domain*

---

**Description**

Returns a Dimension indicated by index for the given TileDB Domain

**Usage**

```
tiledb_domain_get_dimension_from_index(domain, idx)
```

**Arguments**

<code>domain</code>	TileDB Domain object
<code>idx</code>	Integer index of the selected dimension

**Value**

TileDB Dimension object

---

`tiledb_domain_get_dimension_from_name`*Returns a Dimension indicated by name for the given TileDB Domain*

---

**Description**

Returns a Dimension indicated by name for the given TileDB Domain

**Usage**

```
tiledb_domain_get_dimension_from_name(domain, name)
```

**Arguments**

<code>domain</code>	TileDB Domain object
<code>name</code>	A character variable with a dimension name

**Value**

TileDB Dimension object

---

tiledb\_domain\_has\_dimension  
*Check a domain for a given dimension name*

---

**Description**

Check a domain for a given dimension name

**Usage**

```
tiledb_domain_has_dimension(domain, name)
```

**Arguments**

domain	A domain of a TileDB Array schema
name	A character variable with a dimension name

**Value**

A boolean value indicating if the dimension exists in the domain

---

tiledb\_error\_message *Return the error message for a given context*

---

**Description**

Note that this function requires an actual error to have occurred.

**Usage**

```
tiledb_error_message(ctx = tiledb_get_context())
```

**Arguments**

ctx	A tiledb_ctx object
-----	---------------------

**Value**

A character variable with the error message

---

tiledb\_filestore\_buffer\_export

*Export from a TileDB Filestore to a character variable*

---

### Description

Export from a TileDB Filestore to a character variable

### Usage

```
tiledb_filestore_buffer_export(  
    filestore_uri,  
    offset,  
    bytes,  
    ctx = tiledb_get_context()  
)
```

### Arguments

filestore_uri	Character with an TileDB Array Schema URI
offset	(optional) Numeric variable with offset from beginning, default is zero
bytes	(optional) Numeric variable with number of bytes to read, default is zero
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

### Value

A character variable containing the filestore content (subject to offset and bytes) is returned

---

tiledb\_filestore\_buffer\_import

*Import size bytes from a string into a TileDB Filestore*

---

### Description

Import size bytes from a string into a TileDB Filestore

### Usage

```
tiledb_filestore_buffer_import(  
    filestore_uri,  
    buf,  
    bytes,  
    ctx = tiledb_get_context()  
)
```

**Arguments**

filestore_uri	Character with an TileDB Array Schema URI
buf	Character variable with content to be imported
bytes	Number of bytes to be import, defaults to length of buf
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

A boolean is returned to indicate successful completion

---

tiledb\_filestore\_schema\_create

*Create an array schema from a given URI with schema*

---

**Description**

Create an array schema from a given URI with schema

**Usage**

```
tiledb_filestore_schema_create(uri = NULL, ctx = tiledb_get_context())
```

**Arguments**

uri	Character with an TileDB Array Schema URI, if missing or NULL a default schema is returned
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

An ArraySchema object corresponding to the supplied schema, or a default if missing

---

tiledb\_filestore\_size *Return (uncompressed) TileDB Filestore size*

---

**Description**

Return (uncompressed) TileDB Filestore size

**Usage**

```
tiledb_filestore_size(filestore_uri, ctx = tiledb_get_context())
```

**Arguments**

filestore_uri	Character with an TileDB Array Schema URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

A numeric with the size is returned

---

tiledb\_filestore\_uri\_export  
*Export a file from a TileDB Filestore*

---

**Description**

Export a file from a TileDB Filestore

**Usage**

```
tiledb_filestore_uri_export(  
  file_uri,  
  filestore_uri,  
  ctx = tiledb_get_context()  
)
```

**Arguments**

file_uri	Character with a file URI
filestore_uri	Character with an TileDB Array Schema URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

A boolean is returned to indicate successful completion

tiledb\_filestore\_uri\_import  
*Import a file into a TileDB Filestore*

**Description**

Import a file into a TileDB Filestore

**Usage**

```
tiledb_filestore_uri_import(
  filestore_uri,
  file_uri,
  ctx = tiledb_get_context()
)
```

**Arguments**

filestore_uri	Character with an TileDB Array Schema URI
file_uri	Character with a file URI
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

A boolean is returned to indicate successful completion

tiledb\_filter            *Constructs a tiledb\_filter object*

**Description**

Available filters:

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

- "BYTESHUFFLE"
- "POSITIVE\_DELTA"
- "CHECKSUM\_MD5"
- "CHECKSUM\_SHA256"
- "DICTIONARY"
- "SCALE\_FLOAT" (TileDB 2.11.0 or later)
- "FILTER\_XOR" (TileDB 2.12.0 or later)

### 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     *Constructs a tiledb\_filter\_list object*

---

**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

*An S4 class for a TileDB filter list*

---

**Description**

An S4 class for a TileDB filter list

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb\_filter\_set\_option

*Set the option for a filter*

---

**Description**

Set the option for a filter

**Usage**

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

**Arguments**

object	tiledb_filter
option	string
value	int

**Value**

The modified filter object is returned.

**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\_fragment\_info     *Constructs a tiledb\_fragment\_info object*

---

**Description**

Constructs a tiledb\_fragment\_info object

**Usage**

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

**Arguments**

uri                    an character variable with the URI of the array for which fragment info is request  
ctx                    tiledb\_ctx object (optional)

**Value**

tiledb\_fragment\_info object

---

tiledb\_fragment\_info-class

*An S4 class for a TileDB fragment info object*

---

**Description**

An S4 class for a TileDB fragment info object

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb\_fragment\_info\_dense

*Return if a fragment info index is dense*

---

**Description**

Return if a fragment info index is dense

**Usage**

tiledb\_fragment\_info\_dense(object, fid)

**Arguments**

object A TileDB fragment info object

fid A fragment object index

**Value**

A logical value indicating if the fragment is dense

---

`tiledb_fragment_info_dump`*Dump the fragment info to console*

---

**Description**

Dump the fragment info to console

**Usage**

```
tiledb_fragment_info_dump(object)
```

**Arguments**

object            A TileDB fragment info object

**Value**

Nothing is returned, as a side effect the fragment info is displayed

---

`tiledb_fragment_info_get_cell_num`*Return a fragment info number of cells for a given fragment index*

---

**Description**

Return a fragment info number of cells for a given fragment index

**Usage**

```
tiledb_fragment_info_get_cell_num(object, fid)
```

**Arguments**

object            A TileDB fragment info object

fid               A fragment object index

**Value**

A numeric value with the number of cells

---

tiledb\_fragment\_info\_get\_non\_empty\_domain\_index

*Return a fragment info non-empty domain from index*

---

### Description

TODO: Rework with type information

### Usage

tiledb\_fragment\_info\_get\_non\_empty\_domain\_index(object, fid, did, typestr)

### Arguments

object	A TileDB fragment info object
fid	A fragment object index
did	A domain index
typestr	An optional character variable describing the data type which will be accessed from the schema if missing

### Value

A TileDB Domain object

---

tiledb\_fragment\_info\_get\_non\_empty\_domain\_name

*Return a fragment info non-empty domain from name*

---

### Description

TODO: Rework with type information

### Usage

tiledb\_fragment\_info\_get\_non\_empty\_domain\_name(object, fid, dim\_name, typestr)

### Arguments

object	A TileDB fragment info object
fid	A fragment object index
dim_name	A character variable with the dimension name
typestr	An optional character variable describing the data type which will be accessed from the schema if missing

**Value**

A TileDB Domain object

tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_index

*Return a fragment info non-empty domain variable from index*

**Description**

Return a fragment info non-empty domain variable from index

**Usage**

tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_index(object, fid, did)

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index
did	A domain index

**Value**

A character vector with two elements

tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_name

*Return a fragment info non-empty domain variable from name*

**Description**

Return a fragment info non-empty domain variable from name

**Usage**

tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_name(object, fid, dim\_name)

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index
dim_name	A character variable with the dimension name

**Value**

A character vector with two elements

---

`tiledb_fragment_info_get_num`*Return a fragment info number of fragments*

---

**Description**

Return a fragment info number of fragments

**Usage**

```
tiledb_fragment_info_get_num(object)
```

**Arguments**

object            A TileDB fragment info object

**Value**

A numeric variable with the number of fragments

---

`tiledb_fragment_info_get_size`*Return a fragment info fragment size for a given fragment index*

---

**Description**

Return a fragment info fragment size for a given fragment index

**Usage**

```
tiledb_fragment_info_get_size(object, fid)
```

**Arguments**

object            A TileDB fragment info object

fid                A fragment object index

**Value**

A numeric variable with the number of fragments

tiledb\_fragment\_info\_get\_timestamp\_range

*Return a fragment info timestamp range for a given fragment index*

**Description**

Return a fragment info timestamp range for a given fragment index

**Usage**

tiledb\_fragment\_info\_get\_timestamp\_range(object, fid)

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index

**Value**

A Datetime vector with two elements for the range

tiledb\_fragment\_info\_get\_to\_vacuum\_num

*Return the number of fragment info elements to be vacuumed*

**Description**

Return the number of fragment info elements to be vacuumed

**Usage**

tiledb\_fragment\_info\_get\_to\_vacuum\_num(object)

**Arguments**

object	A TileDB fragment info object
--------	-------------------------------

**Value**

A numeric value with the number of to be vacuumed fragments

---

`tiledb_fragment_info_get_to_vacuum_uri`*Return fragment info URI of the to be vacuumed index*

---

**Description**

Return fragment info URI of the to be vacuumed index

**Usage**

```
tiledb_fragment_info_get_to_vacuum_uri(object, fid)
```

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index

**Value**

A character variable with the URI of the be vacuumed index

---

`tiledb_fragment_info_get_unconsolidated_metadata_num`*Return fragment info number of unconsolidated metadata*

---

**Description**

Return fragment info number of unconsolidated metadata

**Usage**

```
tiledb_fragment_info_get_unconsolidated_metadata_num(object)
```

**Arguments**

object	A TileDB fragment info object
--------	-------------------------------

**Value**

A numeric value with the number of unconsolidated metadata

---

`tiledb_fragment_info_get_version`*Return a fragment info version for a given fragment index*

---

**Description**

Return a fragment info version for a given fragment index

**Usage**

```
tiledb_fragment_info_get_version(object, fid)
```

**Arguments**

<code>object</code>	A TileDB fragment info object
<code>fid</code>	A fragment object index

**Value**

A integer value value with the version

---

`tiledb_fragment_info_has_consolidated_metadata`*Return if a fragment info index has consolidated metadata*

---

**Description**

Return if a fragment info index has consolidated metadata

**Usage**

```
tiledb_fragment_info_has_consolidated_metadata(object, fid)
```

**Arguments**

<code>object</code>	A TileDB fragment info object
<code>fid</code>	A fragment object index

**Value**

A logical value indicating consolidated metadata

---

tiledb\_fragment\_info\_sparse

*Return if a fragment info index is sparse*

---

**Description**

Return if a fragment info index is sparse

**Usage**

tiledb\_fragment\_info\_sparse(object, fid)

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index

**Value**

A logical value indicating if the fragment is sparse

---

tiledb\_fragment\_info\_uri

*Return a fragment info URI given its index*

---

**Description**

Return a fragment info URI given its index

**Usage**

tiledb\_fragment\_info\_uri(object, fid)

**Arguments**

object	A TileDB fragment info object
fid	A fragment object index

**Value**

A character variable with URI

---

`tiledb_get_all_metadata`*Return all TileDB Array Metadata objects as a named list*

---

**Description**

Return all TileDB Array Metadata objects as a named list

**Usage**

```
tiledb_get_all_metadata(arr)
```

**Arguments**

`arr`            A TileDB Array object

**Value**

A named list with all Metadata objects indexed by 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     *Return a TileDB Array Metadata object given by key*

---

**Description**

Return a TileDB Array Metadata object given by key

**Usage**

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

**Arguments**

arr	A TileDB Array object
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\_get\_query\_status  
*Retrieve the cached status of the last finalized query*

---

**Description**

This function accesses the status of the last query without requiring the query object.

**Usage**

```
tiledb_get_query_status()
```

**Value**

The status of the last query

---

tiledb_get_vfs	<i>Retrieve a TileDB VFS object from the package environment and cache</i>
----------------	--

---

**Description**

Retrieve a TileDB VFS object from the package environment and cache

**Usage**

```
tiledb_get_vfs()
```

**Value**

A TileDB VFS object

---

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

---

**Description**

Creates a 'tiledb\_group' object

**Usage**

```
tiledb_group(
  uri,
  type = c("READ", "WRITE"),
  ctx = tiledb_get_context(),
  cfg = NULL
)
```

**Arguments**

uri	Character variable with the URI of the new group object
type	Character variable with the query type value: one of "READ" or "WRITE"
ctx	(optional) A TileDB Context object; if not supplied the default context object is retrieved
cfg	(optional) A TileConfig object

**Value**

A 'group' object

---

tiledb_group-class	<i>An S4 class for a TileDB Group object</i>
--------------------	--

---

**Description**

An S4 class for a TileDB Group object

**Slots**

ptr An external pointer to the underlying implementation

---

tiledb_group_add_member	<i>Add Member to TileDB Group</i>
-------------------------	-----------------------------------

---

**Description**

Add Member to TileDB Group

**Usage**

```
tiledb_group_add_member(grp, uri, relative, name = NULL)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
uri	A character value with a new URI
relative	A logical value indicating whether URI is relative to the group
name	An optional character providing a name for the object, defaults to NULL

**Value**

The TileDB Group object, invisibly

---

tiledb\_group\_close      *Close a TileDB Group*

---

**Description**

Close a TileDB Group

**Usage**

```
tiledb_group_close(grp)
```

**Arguments**

grp                      A TileDB Group object as for example returned by tiledb\_group()

**Value**

The TileDB Group object but closed for reading or writing

---

tiledb\_group\_create      *Create a TileDB Group at the given path*

---

**Description**

Create a TileDB Group at the given path

**Usage**

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

**Arguments**

uri                      Character variable with the URI of the new group

ctx                      (optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

The uri path, invisibly

**Examples**

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

---

tiledb\_group\_delete\_metadata

*Deletes Metadata from a TileDB Group*

---

**Description**

Deletes Metadata from a TileDB Group

**Usage**

```
tiledb_group_delete_metadata(grp, key)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with they index under which the data will be written

**Value**

The TileDB Group object, invisibly

---

tiledb\_group\_get\_all\_metadata

*Return all Metadata from a TileDB Group*

---

**Description**

Return all Metadata from a TileDB Group

**Usage**

```
tiledb_group_get_all_metadata(grp)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
-----	---

**Value**

A named List with all Metadata objects index

---

`tiledb_group_get_config`

*Get a TileDB Config from a TileDB Group*

---

**Description**

Get a TileDB Config from a TileDB Group

**Usage**

```
tiledb_group_get_config(grp)
```

**Arguments**

`grp` A TileDB Group object as for example returned by `tiledb_group()`

**Value**

The TileDB Config object of the TileDB Group object

---

`tiledb_group_get_metadata`

*Accesses Metadata from a TileDB Group*

---

**Description**

Accesses Metadata from a TileDB Group

**Usage**

```
tiledb_group_get_metadata(grp, key)
```

**Arguments**

`grp` A TileDB Group object as for example returned by `tiledb_group()`  
`key` A character value with the key of the metadata object to be retrieved

**Value**

The requested object, or NULL is not found

tiledb\_group\_get\_metadata\_from\_index

*Accesses Metadata by Index from a TileDB Group*

---

**Description**

Accesses Metadata by Index from a TileDB Group

**Usage**

```
tiledb_group_get_metadata_from_index(grp, idx)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
idx	A numeric value with the index of the metadata object to be retrieved

**Value**

The requested object, or NULL is not found

---

tiledb\_group\_has\_metadata

*Checks for Metadata in a TileDB Group*

---

**Description**

Checks for Metadata in a TileDB Group

**Usage**

```
tiledb_group_has_metadata(grp, key)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with they index under which the data will be written

**Value**

A boolean value indicating with the object is present

---

tiledb\_group\_is\_open    *Test if TileDB Group is open*

---

**Description**

Test if TileDB Group is open

**Usage**

```
tiledb_group_is_open(grp)
```

**Arguments**

grp                    A TileDB Group object as for example returned by tiledb\_group()

**Value**

A boolean indicating whether the TileDB Group object is open

---

tiledb\_group\_is\_relative  
                          *Test if a Named Group is Using a Relative URI*

---

**Description**

Test if a Named Group is Using a Relative URI

**Usage**

```
tiledb_group_is_relative(grp, name)
```

**Arguments**

grp                    A TileDB Group object as for example returned by tiledb\_group()

name                   A character value with a group name

**Value**

A boolean indicating whether the group uses a relative URI or not

---

tiledb\_group\_member     *Get a Member (Description) by Index from TileDB Group*

---

**Description**

This function returns a three-element character vector with the member object translated to character, uri, and optional name.

**Usage**

```
tiledb_group_member(grp, idx)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
idx	A numeric value with the index of the metadata object to be retrieved

**Value**

A character vector with three elements: the member type, its uri, and name (or "" if the member is unnamed).

---

tiledb\_group\_member\_count  
                          *Get Member Count from TileDB Group*

---

**Description**

Get Member Count from TileDB Group

**Usage**

```
tiledb_group_member_count(grp)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
-----	---

**Value**

The Count of Members in the TileDB Group object

---

`tiledb_group_member_dump`*Dump the TileDB Group to String*

---

**Description**

Dump the TileDB Group to String

**Usage**

```
tiledb_group_member_dump(grp, recursive = FALSE)
```

**Arguments**

<code>grp</code>	A TileDB Group object as for example returned by <code>tiledb_group()</code>
<code>recursive</code>	A logical value indicating whether a recursive dump is desired, defaults to 'FALSE'. Note that recursive listings on remote object may be an expensive or slow operation.

**Value**

A character string

---

`tiledb_group_metadata_num`*Returns Number of Metadata Objects a TileDB Group*

---

**Description**

Returns Number of Metadata Objects a TileDB Group

**Usage**

```
tiledb_group_metadata_num(grp)
```

**Arguments**

<code>grp</code>	A TileDB Group object as for example returned by <code>tiledb_group()</code>
------------------	--

**Value**

A numeric value with the number of metadata objects

---

tiledb_group_open	<i>Open a TileDB Group</i>
-------------------	----------------------------

---

**Description**

Open a TileDB Group

**Usage**

```
tiledb_group_open(grp, type = c("READ", "WRITE"))
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
type	A character value that must be either 'READ' or 'WRITE'

**Value**

The TileDB Group object but opened for reading or writing

---

tiledb_group_put_metadata	<i>Write Metadata to a TileDB Group</i>
---------------------------	---

---

**Description**

Write Metadata to a TileDB Group

**Usage**

```
tiledb_group_put_metadata(grp, key, val)
```

**Arguments**

grp	A TileDB Group object as for example returned by tiledb_group()
key	A character value with they index under which the data will be written
val	An R object (numeric, int, or char vector) that will be stored

**Value**

On success boolean 'TRUE' is returned

---

`tiledb_group_query_type`*Return a TileDB Group query type*

---

**Description**

Return a TileDB Group query type

**Usage**

```
tiledb_group_query_type(grp)
```

**Arguments**

`grp` A TileDB Group object as for example returned by `tiledb_group()`

**Value**

A character value with the query type i.e. one of “READ” or “WRITE”.

---

`tiledb_group_remove_member`*Remove Member from TileDB Group*

---

**Description**

Remove Member from TileDB Group

**Usage**

```
tiledb_group_remove_member(grp, uri)
```

**Arguments**

`grp` A TileDB Group object as for example returned by `tiledb_group()`

`uri` A character value with a the URI of the member to be removed, or (if added with a name) the name of the member

**Value**

The TileDB Group object, invisibly

---

`tiledb_group_set_config`*Set a TileDB Config for a TileDB Group*

---

**Description**

Set a TileDB Config for a TileDB Group

**Usage**

```
tiledb_group_set_config(grp, cfg)
```

**Arguments**

<code>grp</code>	A TileDB Group object as for example returned by <code>tiledb_group()</code>
<code>cfg</code>	A TileDB Config object

**Value**

The TileDB Group object with added Config

---

`tiledb_group_uri`*Return a TileDB Group URI*

---

**Description**

Return a TileDB Group URI

**Usage**

```
tiledb_group_uri(grp)
```

**Arguments**

<code>grp</code>	A TileDB Group object as for example returned by <code>tiledb_group()</code>
------------------	--

**Value**

A character value with the URI

---

tiledb\_has\_metadata    *Test if TileDB Array has Metadata*

---

**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 metadata of the given array

---

tiledb\_is\_supported\_fs  
*Query if a TileDB backend is supported*

---

**Description**

The scheme corresponds to the URI scheme for TileDB resources.

**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     *Return count of TileDB Array Metadata objects*

---

**Description**

Return count of TileDB Array Metadata objects

**Usage**

```
tiledb_num_metadata(arr)
```

**Arguments**

arr                    A TileDB Array object

**Value**

A integer variable with the number of Metadata objects

---

tiledb\_object\_ls         *List TileDB resources at a given root URI path*

---

**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           *Move a TileDB resource to new uri path*

---

**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           *Removes a TileDB resource*

---

**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 to TileDB resource  
 ctx                   tiledb\_ctx object (optional)

**Value**

uri of removed TileDB resource

---

tiledb\_object\_type      *Return the TileDB object type string of a TileDB resource*

---

**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      *Recursively discover TileDB resources at a given root URI path*

---

**Description**

Recursively discover TileDB resources at a given root URI path

**Usage**

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

**Arguments**

uri	root uri path to walk
order	traversal order, one of "PREORDER" and "POSTORDER" (default "PREORDER")
ctx	tiledb_ctx object (optional)

**Value**

a dataframe with object type, object uri string columns

---

tiledb\_put\_metadata     *Store an object in TileDB Array Metadata under given key*

---

**Description**

Store an object in TileDB Array Metadata under given key

**Usage**

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

**Arguments**

arr	A TileDB Array object
key	A character value describing a metadata key
val	An object to be stored

**Value**

A boolean value indicating success

---

tiledb\_query     *Creates a 'tiledb\_query' object*

---

**Description**

Creates a 'tiledb\_query' object

**Usage**

```
tiledb_query(
  array,
  type = if (tiledb_version(TRUE) >= "2.12.0") c("READ", "WRITE", "DELETE",
    "MODIFY_EXCLUSIVE") else c("READ", "WRITE"),
  ctx = tiledb_get_context()
)
```

**Arguments**

array	A TileDB Array object
type	A character value that must be one of 'READ', 'WRITE', or 'DELETE' (for TileDB >= 2.12.0)
ctx	(optional) A TileDB Ctx object

**Value**

'tiledb\_query' object

---

tiledb\_query-class     *An S4 class for a TileDB Query object*

---

**Description**

An S4 class for a TileDB Query object

**Slots**

ptr An external pointer to the underlying implementation

---

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

---

**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, also supplying type*


---

**Description**

Set a range for a given query, also supplying type

**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, nullable = FALSE)
```

**Arguments**

sizeoffsets	A numeric value with the size of the offsets vector
sizedata	A numeric value of the size of the data string
nullable	An optional boolean indicating whether the column can have NULLs

**Value**

An external pointer to the allocated buffer object

---

tiledb\_query\_apply\_aggregate

*Run an aggregate operation on the given query attribute*

---

**Description**

Run an aggregate operation on the given query attribute

**Usage**

```
tiledb_query_apply_aggregate(
  query,
  attrname,
  operation = c("Count", "NullCount", "Min", "Max", "Mean", "Sum"),
  nullable = TRUE
)
```

**Arguments**

query	A TileDB Query object
attrname	The name of an attribute
operation	The name of aggregation operation
nullable	A boolean toggle whether the attribute is nullable

**Value**

The value of the aggregation

---

```
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,
    nullable = FALSE,
    varnum = 1
)
```

**Arguments**

query	A TileDB Query object
datatype	A character value containing the data type
ncells	A number of elements (not bytes)
nullable	Optional boolean parameter indicating whether missing values are allowed (for which another column is allocated), default is FALSE
varnum	Option integer parameter for the number of elements per variable, default is one

**Value**

An external pointer to the allocated buffer object

---

```
tiledb_query_condition
    Creates a 'tiledb_query_condition' object
```

---

**Description**

Creates a 'tiledb\_query\_condition' object

**Usage**

```
tiledb_query_condition(ctx = tiledb_get_context())
```

**Arguments**

ctx (optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

A 'tiledb\_query\_condition' object

---

tiledb\_query\_condition-class

*An S4 class for a TileDB QueryCondition object*

---

**Description**

An S4 class for a TileDB QueryCondition object

**Slots**

ptr An external pointer to the underlying implementation

init A logical variable tracking if the query condition object has been initialized

---

tiledb\_query\_condition\_combine

*Combine two 'tiledb\_query\_condition' objects*

---

**Description**

Combines two query condition object using a relational operator. Support for operator 'AND' is generally available, the 'OR' operator is available if TileDB 2.10 or newer is used.

**Usage**

tiledb\_query\_condition\_combine(lhs, rhs, op)

**Arguments**

lhs A 'tiledb\_query\_condition' object on the left-hand side of the relation

rhs A 'tiledb\_query\_condition' object on the left-hand side of the relation

op A character value with then relation, this must be one of 'AND', 'OR' or 'NOT'.

**Value**

The combined 'tiledb\_query\_condition' object

---

tiledb\_query\_condition\_create

*Create a query condition for vector 'IN' and 'NOT\_IN' operations*


---

### Description

Uses 'IN' and 'NOT\_IN' operators on given attribute

### Usage

```
tiledb_query_condition_create(
    name,
    values,
    op = "IN",
    ctx = tiledb_get_context()
)
```

### Arguments

name	A character value with the scheme attribute name
values	A vector wiith the given values, supported types are integer, double, integer64 and charactor
op	(optional) A character value with the chosen set operation, this must be one of 'IN' or 'NOT_IN'; default to 'IN'
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

### Value

A query condition object is returned

---

tiledb\_query\_condition\_init

*Initialize a 'tiledb\_query\_condition' object*


---

### Description

Initializes (and possibly allocates) a query condition object using a triplet of attribute name, comparison value, and operator. Six types of conditions are supported, they all take a single scalar comparison argument and attribute to compare against. At present only integer or numeric attribute comparisons are implemented.

**Usage**

```
tiledb_query_condition_init(
    attr,
    value,
    dtype,
    op,
    qc = tiledb_query_condition()
)
```

**Arguments**

attr	A character value with the scheme attribute name
value	A scalar value that the attribute is compared against
dtype	A character value with the TileDB data type of the attribute column, for example 'FLOAT64' or 'INT32'
op	A character value with then comparison operation, this must be one of 'LT', 'LE', 'GT', 'GE', 'EQ', 'NE'.
qc	(optional) A 'tiledb_query_condition' object to be initialized by this call, if none is given a new one is allocated.

**Value**

The initialized 'tiledb\_query\_condition' object

---

```
tiledb_query_condition_set_use_enumeration
    Enable use of enumeration in query condition
```

---

**Description**

Set a boolean toggle to signal use of enumeration in query condtion (TileDB 2.17 or later)

**Usage**

```
tiledb_query_condition_set_use_enumeration(
    qc,
    use_enum,
    ctx = tiledb_get_context()
)
```

**Arguments**

qc	A 'tiledb_query_condition' object
use_enum	A boolean to set (if TRUE) or unset (if FALSE) enumeration use
ctx	(optional) A TileDB Ctx object; if not supplied the default context object is retrieved

**Value**

Nothing is returned, the function is invoked for the side effect

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

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

**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_ctx	<i>Return query context object</i>
------------------	------------------------------------

---

**Description**

Return query context object

**Usage**

```
tiledb_query_ctx(query)
```

**Arguments**

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

**Value**

A TileDB Context object retrieved from the query

---

tiledb_query_export_buffer	<i>Export Query Buffer to Pair of Arrow IO Pointers</i>
----------------------------	---

---

**Description**

This function exports the named buffer from a 'READ' query to two Arrow C pointers.

**Usage**

```
tiledb_query_export_buffer(query, name, ctx = tiledb_get_context())
```

**Arguments**

query	A TileDB Query object
name	A character variable identifying the buffer
ctx	tiledb_ctx object (optional)

**Value**

A nanoarrow object (which is an external pointer to an Arrow Array with the Arrow Schema stored as the external pointer tag) classed as an S3 object

---

tiledb\_query\_finalize *Finalize TileDB Query*

---

**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_est_result_size`*Retrieve the estimated result size for a query and attribute*

---

**Description**

When reading from sparse arrays, one cannot know beforehand how big the result will be (unless one actually executes the query). This function offers a way to get the estimated result size for the given attribute. As TileDB does not actually execute the query, getting the estimated result is very fast.

**Usage**

```
tiledb_query_get_est_result_size(query, name)
```

**Arguments**

`query`            A TileDB Query object  
`name`            A variable with an attribute name

**Value**

An estimate of the query result size

---

tiledb\_query\_get\_est\_result\_size\_var

*Retrieve the estimated result size for a query and variable-sized attribute*


---

**Description**

When reading variable-length attributes from either dense or sparse arrays, one cannot know beforehand how big the result will be (unless one actually executes the query). This function offers a way to get the estimated result size for the given attribute. As TileDB does not actually execute the query, getting the estimated result is very fast.

**Usage**

```
tiledb_query_get_est_result_size_var(query, name)
```

**Arguments**

query	A TileDB Query object
name	A variable with an attribute name

**Value**

An estimate of the query result size

---

tiledb\_query\_get\_fragment\_num

*Retrieve the Number of Fragments for Query*


---

**Description**

This function is only applicable to 'WRITE' queries.

**Usage**

```
tiledb_query_get_fragment_num(query)
```

**Arguments**

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

**Value**

An integer with the number of fragments for the given query

---

tiledb\_query\_get\_fragment\_timestamp\_range

*Retrieve the timestamp range for a given Query Fragment*


---

### Description

This function is only applicable to ‘WRITE’ queries. The time resolution in TileDB is milliseconds since the epoch so an R `Datetime` vector is returned.

### Usage

```
tiledb_query_get_fragment_timestamp_range(query, idx)
```

### Arguments

query	A TileDB Query object
idx	An integer (or numeric) index ranging from zero to the number of fragments minus 1

### Value

A two-element datetime vector with the start and end time of the fragment write.

---

tiledb\_query\_get\_fragment\_uri

*Retrieve the URI for a given Query Fragment*


---

### Description

This function is only applicable to ‘WRITE’ queries.

### Usage

```
tiledb_query_get_fragment_uri(query, idx)
```

### Arguments

query	A TileDB Query object
idx	An integer (or numeric) index ranging from zero to the number of fragments minus 1

### Value

An character value with the fragment URI

---

tiledb\_query\_get\_layout  
*Get TileDB Query layout*

---

**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\_get\_range  
*Retrieve the query range for a query dimension and range index*

---

**Description**

Retrieve the query range for a query dimension and range index

**Usage**

```
tiledb_query_get_range(query, dimidx, rngidx)
```

**Arguments**

query            A TileDB Query object  
dimidx           An integer or numeric index selecting the dimension  
rngidx           An integer or numeric index selection the given range for the dimension

**Value**

An integer vector with elements start, end and stride for the query range for the given dimension and range index

---

tiledb\_query\_get\_range\_num

*Retrieve the number of ranges for a query dimension*


---

**Description**

Retrieve the number of ranges for a query dimension

**Usage**

```
tiledb_query_get_range_num(query, idx)
```

**Arguments**

query	A TileDB Query object
idx	An integer or numeric index selecting the dimension

**Value**

An integer with the number of query range for the given dimensions

---

tiledb\_query\_get\_range\_var

*Retrieve the query range for a variable-sized query dimension and range index*


---

**Description**

Retrieve the query range for a variable-sized query dimension and range index

**Usage**

```
tiledb_query_get_range_var(query, dimidx, rngidx)
```

**Arguments**

query	A TileDB Query object
dimidx	An integer index selecting the variable-sized dimension
rngidx	An integer index selection the given range for the dimension

**Value**

An string vector with elements start and end for the query range for the given dimension and range index

---

tiledb\_query\_import\_buffer

*Import to Query Buffer from Pair of Arrow IO Pointers*


---

### Description

This function imports to the named buffer for a 'WRITE' query from two Arrow external pointers.

### Usage

```
tiledb_query_import_buffer(
  query,
  name,
  nanoarrowptr,
  ctx = tiledb_get_context()
)
```

### Arguments

query	A TileDB Query object
name	A character variable identifying the buffer
nanoarrowptr	A nanoarrow object (which is an external pointer to an Arrow Array with the Arrow Schema stored as the external pointer tag) classed as an S3 object
ctx	tiledb_ctx object (optional)

### Value

The update Query external pointer is returned

---

tiledb\_query\_result\_buffer\_elements

*Get TileDB Query result buffer element size*


---

### Description

The underlying library functions returns a pair of values as a vector of length two. The first number is the number of element offsets for variable size attributes (and always zero for fixed-sized attributes and coordinates). The second is the number of elements in the data buffer. For variable-sized attributes the first number is the number of cells read (and hence the number of offsets), the second number is the number of elements in the data buffer.

### Usage

```
tiledb_query_result_buffer_elements(query, attr)
```

**Arguments**

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

**Details**

As this function was first made available when only a scalar (corresponding to the second result) was returned, we still return that value.

**Value**

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

**See Also**

tiledb\_query\_result\_buffer\_elements\_vec

---

tiledb\_query\_result\_buffer\_elements\_vec

*Get TileDB Query result buffer element size pair as vector*

---

**Description**

The underlying library functions returns a pair of values as a vector of length two. The first number is the number of element offsets for variable size attributes (and always zero for fixed-sized attributes and coordinates). The second is the number of elements in the data buffer. For variable-sized attributes the first number is the number of cells read (and hence the number of offsets), the second number is the number of elements in the data buffer. In the case of a nullable attribute, a third element is returned with the size of the validity buffer.

**Usage**

```
tiledb_query_result_buffer_elements_vec(query, attr, nullable = FALSE)
```

**Arguments**

query	A TileDB Query object
attr	A character value containing the attribute
nullable	A logical variable that is 'TRUE' to signal that the attribute is nullable, and 'FALSE' otherwise

**Value**

A vector with the number of elements in the offsets buffer (and zero for fixed-size attribute or dimensions), the number elements in the results buffer for the given attribute, and (if nullable) a third element with the validity buffer size.

**See Also**

tiledb\_query\_result\_buffer\_elements

---

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 and tiledb\_query\_buffer\_assign\_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

*Assign a buffer to a Query attribute*

---

### **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\_condition

*Set a query combination object for a query*

---

### **Description**

Set a query combination object for a query

### **Usage**

tiledb\_query\_set\_condition(query, qc)

### **Arguments**

query	A TileDB Query object
qc	A TileDB Query Combination object

### **Value**

The modified query object, invisibly

---

tiledb\_query\_set\_layout  
*Set TileDB Query layout*

---

**Description**

Set TileDB Query layout

**Usage**

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

**Arguments**

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

**Value**

The modified query object, invisibly

---

tiledb\_query\_set\_subarray  
*Set subarray for TileDB Query object*

---

**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\_stats      *Return query statistics as a JSON string*

---

**Description**

Return query statistics as a JSON string

**Usage**

```
tiledb_query_stats(query)
```

**Arguments**

query                  A TileDB Query object

**Value**

A JSON-formatted string with context statistics

---

tiledb\_query\_status      *Get TileDB Query status*

---

**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     *Submit TileDB Query*

---

**Description**

Note that the query object may need to be finalized via tiledb\_query\_finalize.

**Usage**

```
tiledb_query_submit(query)
```

**Arguments**

query             A TileDB Query object

**Value**

The modified query object, invisibly

---

tiledb\_query\_submit\_async  
*Submit TileDB Query asynchronously without a callback returning immediately*

---

**Description**

Note that the query object may need to be finalized via tiledb\_query\_finalize.

**Usage**

```
tiledb_query_submit_async(query)
```

**Arguments**

query             A TileDB Query object

**Value**

The modified query object, invisibly

tiledb\_query\_type      *Return TileDB Query type*

---

**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\_dim\_attr\_status  
*Get Dimension or Attribute Status*

---

**Description**

Note that this function is an unexported internal function.

**Usage**

tiledb\_schema\_get\_dim\_attr\_status(sch)

**Arguments**

sch                    A TileDB Schema object

**Value**

An integer vector where each element corresponds to a schema entry, and a value of one signals dimension and a value of two an attribute.

---

tiledb\_schema\_get\_enumeration\_status  
*Get Dimension or Attribute Status*

---

**Description**

Note that this function is an unexported internal function.

**Usage**

```
tiledb_schema_get_enumeration_status(sch)
```

**Arguments**

sch                    A TileDB Schema object

**Value**

An integer vector where each element corresponds to a schema entry, and a value of one signals dimension and a value of two an attribute.

---

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_schema_object`    *Succinctly describe a TileDB array schema*

---

**Description**

This is an internal function that is not exported.

**Usage**

```
tiledb_schema_object(array)
```

**Arguments**

`array`                    A TileDB Array object

**Value**

A list containing two data frames, one describing the overall array as well as one with descriptions about dimensions and attributes in the schema

---

tiledb\_set\_context      *Store a TileDB context object in the package cache*

---

**Description**

Store a TileDB context object in the package cache

**Usage**

```
tiledb_set_context(ctx)
```

**Arguments**

ctx                    A TileDB context object

**Value**

NULL, invisibly. The function is invoked for the side-effect of storing the VFS object.

---

tiledb\_set\_vfs            *Store a TileDB VFS object in the package environment*

---

**Description**

Store a TileDB VFS object in the package environment

**Usage**

```
tiledb_set_vfs(vfs)
```

**Arguments**

vfs                    A TileDB VFS object

**Value**

NULL, invisibly. The function is invoked for the side-effect of storing the VFS object.

---

tiledb\_stats\_disable    *Disable internal TileDB statistics counters*

---

**Description**

This function ends the collection of internal statistics.

**Usage**

```
tiledb_stats_disable()
```

---

tiledb\_stats\_dump        *Dumps internal TileDB statistics to file or stdout*

---

**Description**

Dumps internal TileDB statistics to file or stdout

**Usage**

```
tiledb_stats_dump(path)
```

**Arguments**

path                    Character variable with path to stats file; if the empty string is passed then the result is displayed on stdout.

**Examples**

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

---

tiledb\_stats\_enable    *Enable internal TileDB statistics counters*

---

**Description**

This function starts the collection of internal statistics.

**Usage**

```
tiledb_stats_enable()
```

---

tiledb\_stats\_print     *Print internal TileDB statistics*

---

**Description**

This function is a convenience wrapper for tiledb\_stats\_dump.

**Usage**

```
tiledb_stats_print()
```

---

tiledb\_stats\_raw\_dump     *Dumps internal TileDB statistics as JSON to a string*

---

**Description**

This function requires TileDB Embedded 2.0.3 or later.

**Usage**

```
tiledb_stats_raw_dump()
```

**Examples**

```
txt <- tiledb_stats_raw_dump()
cat(txt, "\n")
```

---

tiledb\_stats\_raw\_get     *Gets internal TileDB statistics as JSON string*

---

**Description**

This function is a (now deprecated) convenience wrapper for tiledb\_stats\_raw\_dump and returns the result as a JSON string. It required TileDB Embedded 2.0.3 or later.

**Usage**

```
tiledb_stats_raw_get()
```

---

`tiledb_stats_raw_print`*Print internal TileDB statistics as JSON*

---

**Description**

This function is a convenience wrapper for `tiledb_stats_raw_dump`. It required TileDB Embedded 2.0.3 or later.

**Usage**

```
tiledb_stats_raw_print()
```

---

`tiledb_stats_reset`*Reset internal TileDB statistics counters*

---

**Description**

This function resets the counters for internal statistics.

**Usage**

```
tiledb_stats_reset()
```

---

`tiledb_subarray`*Constructs a tiledb\_subarray object from a TileDB Query*

---

**Description**

Constructs a `tiledb_subarray` object from a TileDB Query

**Usage**

```
tiledb_subarray(query)
```

**Arguments**

`query`            A TileDB Query Object

**Value**

`tiledb_subarray` object

---

tiledb\_subarray-class *An S4 class for a TileDB Subarray*

---

**Description**

An S4 class for a TileDB Subarray

**Slots**

ptr External pointer to the underlying implementation

---

tiledb\_subarray\_to\_query  
*Apply a Subarray to a Query*

---

**Description**

Apply a Subarray to a Query

**Usage**

```
tiledb_subarray_to_query(query, subarray)
```

**Arguments**

query	A TileDB Query Object
subarray	A TileDB Subarray Object

**Value**

tiledb\_query object

---

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**

The 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_close	<i>Close a TileDB VFS Filehandle</i>
------------------	--------------------------------------

---

**Description**

Close a TileDB VFS Filehandle

**Usage**

```
tiledb_vfs_close(fh, ctx = tiledb_get_context())
```

**Arguments**

fh	A TileDB VFS Filehandle external pointer as returned from tiledb_vfs_open
ctx	(optional) A TileDB Ctx object

**Value**

The result of the close operation is returned.

---

tiledb\_vfs\_copy\_file *Copy a file to VFS*

---

**Description**

Copy a file to VFS

**Usage**

```
tiledb_vfs_copy_file(file, uri, vfs = tiledb_get_vfs())
```

**Arguments**

file	Character variable with a local file path
uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value of the removed file

---

tiledb\_vfs\_create\_bucket  
*Create a VFS Bucket*

---

**Description**

Create a VFS Bucket

**Usage**

```
tiledb_vfs_create_bucket(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value

---

tiledb\_vfs\_create\_dir *Create a VFS Directory*

---

**Description**

Create a VFS Directory

**Usage**

```
tiledb_vfs_create_dir(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a directory path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value of the created directory

---

tiledb\_vfs\_dir\_size *Return VFS Directory Size*

---

**Description**

Return VFS Directory Size

**Usage**

```
tiledb_vfs_dir_size(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The size of the directory

---

`tiledb_vfs_empty_bucket`*Empty a VFS Bucket*

---

**Description**

Empty a VFS Bucket

**Usage**

```
tiledb_vfs_empty_bucket(uri, vfs = tiledb_get_vfs())
```

**Arguments**

<code>uri</code>	Character variable with a URI describing a cloud bucket
<code>vfs</code>	A TileDB VFS object; default is to use a cached value.

**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(uri, vfs = tiledb_get_vfs())
```

**Arguments**

<code>uri</code>	Character variable with a URI describing a file path
<code>vfs</code>	A TileDB VFS object; default is to use a cached value.

**Value**

The size of the file

---

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

---

**Description**

Check for VFS Bucket

**Usage**

```
tiledb_vfs_is_bucket(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

**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(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a directory path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

A boolean value indicating if it is a directory

---

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

---

**Description**

Check for empty VFS Bucket

**Usage**

```
tiledb_vfs_is_empty_bucket(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

**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      *Test for VFS File*

---

**Description**

Test for VFS File

**Usage**

```
tiledb_vfs_is_file(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri                      Character variable with a URI describing a file path  
vfs                      A TileDB VFS object; default is to use a cached value.

**Value**

A boolean value indicating if it is a file

---

tiledb\_vfs\_ls              *Return VFS Directory Listing*

---

**Description**

Return VFS Directory Listing

**Usage**

```
tiledb_vfs_ls(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri                      Character variable with a URI describing a file path  
vfs                      A TileDB VFS object; default is to use a cached value.

**Value**

The content of the directory, non-recursive

---

tiledb\_vfs\_ls\_recursive

*Recursively list objects from given URI*


---

### Description

This functionality is currently limited to S3 URIs.

### Usage

```
tiledb_vfs_ls_recursive(
  uri,
  vfs = tiledb_get_vfs(),
  ctx = tiledb_get_context()
)
```

### Arguments

uri	Character variable with a URI describing a file path
vfs	(optiona) A TileDB VFS object; default is to use a cached value.
ctx	(optional) A TileDB Ctx object

### Value

A data.frame object with two columns for the full path and the object size in bytes

---

tiledb\_vfs\_move\_dir *Move (or rename) a VFS Directory*


---

### Description

Move (or rename) a VFS Directory

### Usage

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

### Arguments

olduri	Character variable with an existing URI describing a directory path
newuri	Character variable with a new desired URI directory path
vfs	A TileDB VFS object; default is to use a cached value.

### 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(olduri, newuri, vfs = tiledb_get_vfs())
```

**Arguments**

olduri	Character variable with an existing URI describing a file path
newuri	Character variable with a new desired URI file path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The newuri value of the moved file

---

tiledb\_vfs\_open        *Open a TileDB VFS Filehandle for reading or writing*

---

**Description**

Open a TileDB VFS Filehandle for reading or writing

**Usage**

```
tiledb_vfs_open(
  binfile,
  mode = c("READ", "WRITE", "APPEND"),
  vfs = tiledb_get_vfs(),
  ctx = tiledb_get_context()
)
```

**Arguments**

binfile	A character variable describing the (binary) file to be opened
mode	A character variable with value 'READ', 'WRITE' or 'APPEND'
vfs	A TileDB VFS object; default is to use a cached value.
ctx	(optional) A TileDB Ctx object

**Value**

A TileDB VFS Filehandle object (as an external pointer)

---

tiledb_vfs_read	<i>Read from a TileDB VFS Filehandle</i>
-----------------	--

---

**Description**

This interface currently defaults to reading an integer vector. This is suitable for R objects as a raw vector used for (de)serialization can be mapped easily to an integer vector. It is also possible to memcpy to the contiguous memory of an integer vector should other (non-R) data be transferred.

**Usage**

```
tiledb_vfs_read(fh, offset, nbytes, ctx = tiledb_get_context())
```

**Arguments**

fh	A TileDB VFS Filehandle external pointer as returned from tiledb_vfs_open
offset	A scalar value with the byte offset from the beginning of the file with a of zero.
nbytes	A scalar value with the number of bytes to be read.
ctx	(optional) A TileDB Ctx object

**Value**

The binary file content is returned as an integer vector.

---

tiledb_vfs_remove_bucket	<i>Remove a VFS Bucket</i>
--------------------------	----------------------------

---

**Description**

Remove a VFS Bucket

**Usage**

```
tiledb_vfs_remove_bucket(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a cloud bucket
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value

---

tiledb\_vfs\_remove\_dir *Remove a VFS Directory*

---

**Description**

Remove a VFS Directory

**Usage**

```
tiledb_vfs_remove_dir(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a directory path
vfs	A TileDB VFS object; default is to use a cached value.

**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(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a file path
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value of the removed file

---

`tiledb_vfs_serialize` *Serialize an R Object to a VFS-accessible URI*

---

**Description**

Serialize an R Object to a VFS-accessible URI

**Usage**

```
tiledb_vfs_serialize(obj, uri, vfs = tiledb_get_vfs())
```

**Arguments**

<code>obj</code>	An R object which will be passed to <code>serialize()</code>
<code>uri</code>	Character variable with a URI describing a file path to an RDS file
<code>vfs</code>	A TileDB VFS object; default is to use a cached value.

**Value**

The uri is returned invisibly

---

`tiledb_vfs_sync` *Sync a TileDB VFS Filehandle*

---

**Description**

Sync a TileDB VFS Filehandle

**Usage**

```
tiledb_vfs_sync(fh, ctx = tiledb_get_context())
```

**Arguments**

<code>fh</code>	A TileDB VFS Filehandle external pointer as returned from <code>tiledb_vfs_open</code>
<code>ctx</code>	(optional) A TileDB Ctx object

**Value**

The result of the sync operation is returned.

---

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

---

**Description**

Touch a VFS URI Resource

**Usage**

```
tiledb_vfs_touch(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a bucket, file or directory
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The uri value

---

tiledb_vfs_unserialize	<i>Unserialize an R Object from a VFS-accessible URI</i>
------------------------	--

---

**Description**

Unserialize an R Object from a VFS-accessible URI

**Usage**

```
tiledb_vfs_unserialize(uri, vfs = tiledb_get_vfs())
```

**Arguments**

uri	Character variable with a URI describing a file path to an RDS file
vfs	A TileDB VFS object; default is to use a cached value.

**Value**

The unserialized object

---

tiledb_vfs_write	<i>Write to a TileDB VFS Filehandle</i>
------------------	---

---

**Description**

This interface currently defaults to using an integer vector. This is suitable for R objects as the raw vector result from serialization can be mapped easily to an integer vector. It is also possible to memcopy to the contiguous memory of an integer vector should other (non-R) data be transferred.

**Usage**

```
tiledb_vfs_write(fh, vec, ctx = tiledb_get_context())
```

**Arguments**

fh	A TileDB VFS Filehandle external pointer as returned from tiledb_vfs_open
vec	An integer vector of content to be written
ctx	(optional) A TileDB Ctx object

**Value**

The result of the write operation is returned.

---

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

### Value

The resulting elements in the selected format

---

[, 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_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\_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"]
```

# Index

[, tiledb\_array  
 ([, tiledb\_array, ANY-method),  
 191

[, tiledb\_array, ANY, ANY, tiledb\_array-method  
 ([, tiledb\_array, ANY-method),  
 191

[, tiledb\_array, ANY, tiledb\_array-method  
 ([, tiledb\_array, ANY-method),  
 191

[, tiledb\_array, ANY-method, 191

[, tiledb\_array-method  
 ([, tiledb\_array, ANY-method),  
 191

[, tiledb\_config  
 ([, tiledb\_config, ANY-method),  
 191

[, tiledb\_config, ANY, ANY, tiledb\_config-method  
 ([, tiledb\_config, ANY-method),  
 191

[, tiledb\_config, ANY, tiledb\_config-method  
 ([, tiledb\_config, ANY-method),  
 191

[, tiledb\_config, ANY-method, 191

[, tiledb\_config-method  
 ([, tiledb\_config, ANY-method),  
 191

[, tiledb\_filter\_list  
 ([, tiledb\_filter\_list, ANY-method),  
 192

[, tiledb\_filter\_list, ANY, ANY, tiledb\_filter\_list-method  
 ([, tiledb\_filter\_list, ANY-method),  
 192

[, tiledb\_filter\_list, ANY, tiledb\_filter\_list-method  
 ([, tiledb\_filter\_list, ANY-method),  
 192

[, tiledb\_filter\_list, ANY-method, 192

[, tiledb\_filter\_list-method  
 ([, tiledb\_filter\_list, ANY-method),  
 192

[<- , tiledb\_array, ANY, ANY, ANY-method,  
 193

[<- , tiledb\_config, ANY, ANY, ANY-method,  
 194

[<- , tiledb\_array  
 ([<- , tiledb\_array, ANY, ANY, ANY-method),  
 193

[<- , tiledb\_array, ANY, ANY, tiledb\_array-method  
 ([<- , tiledb\_array, ANY, ANY, ANY-method),  
 193

[<- , tiledb\_array, ANY, tiledb\_array-method  
 ([<- , tiledb\_array, ANY, ANY, ANY-method),  
 193

[<- , tiledb\_array-method  
 ([<- , tiledb\_array, ANY, ANY, ANY-method),  
 193

[<- , tiledb\_config  
 ([<- , tiledb\_config, ANY, ANY, ANY-method),  
 194

[<- , tiledb\_config, ANY, ANY, tiledb\_config-method  
 ([<- , tiledb\_config, ANY, ANY, ANY-method),  
 194

[<- , tiledb\_config, ANY, tiledb\_config-method  
 ([<- , tiledb\_config, ANY, ANY, ANY-method),  
 194

[<- , tiledb\_config-method  
 ([<- , tiledb\_config, ANY, ANY, ANY-method),  
 194

allows\_dups, 9

allows\_dups, tiledb\_array\_schema-method  
 (allows\_dups), 9

allows\_dups<-, 10

allows\_dups<-, tiledb\_array\_schema-method  
 (allows\_dups<-), 10

array\_consolidate, 10

array\_vacuum, 11

as.data.frame.tiledb\_config, 12

as.vector.tiledb\_config, 12

attrs (generics), 37

- attrs, tiledb\_array, ANY-method, [13](#)
- attrs, tiledb\_array\_schema, ANY-method, [13](#)
- attrs, tiledb\_array\_schema, character-method, [14](#)
- attrs, tiledb\_array\_schema, numeric-method, [15](#)
- attrs<-, tiledb\_array-method, [16](#)
- attrs<- (generics), [37](#)
- capacity, [16](#)
- capacity, tiledb\_array\_schema-method (capacity), [16](#)
- capacity<-, [17](#)
- capacity<-, tiledb\_array\_schema-method (capacity<-), [17](#)
- cell\_order (generics), [37](#)
- cell\_order, tiledb\_array\_schema-method, [17](#)
- cell\_val\_num, [18](#)
- cell\_val\_num, tiledb\_attr-method (cell\_val\_num), [18](#)
- cell\_val\_num, tiledb\_dim-method, [18](#)
- cell\_val\_num<-, [19](#)
- cell\_val\_num<-, tiledb\_attr-method (cell\_val\_num<-), [19](#)
- check (schema\_check), [59](#)
- check, tiledb\_array\_schema-method (schema\_check), [59](#)
- completedBatched, [19](#)
- config (generics), [37](#)
- config, tiledb\_ctx-method, [20](#)
- createBatched, [21](#)
- datatype (generics), [37](#)
- datatype, tiledb\_attr-method, [21](#)
- datatype, tiledb\_dim-method, [22](#)
- datatype, tiledb\_domain-method, [22](#)
- datetimes\_as\_int64, [23](#)
- datetimes\_as\_int64, tiledb\_array-method (datetimes\_as\_int64), [23](#)
- datetimes\_as\_int64<-, [24](#)
- datetimes\_as\_int64<-, tiledb\_array-method (datetimes\_as\_int64<-), [24](#)
- describe, [24](#)
- dim.tiledb\_array\_schema, [25](#)
- dim.tiledb\_dim, [25](#)
- dim.tiledb\_domain, [26](#)
- dimensions (generics), [37](#)
- dimensions, tiledb\_array\_schema-method, [27](#)
- dimensions, tiledb\_domain-method, [27](#)
- domain (generics), [37](#)
- domain, tiledb\_array\_schema-method, [28](#)
- domain, tiledb\_dim-method, [29](#)
- extended, [29](#)
- extended, tiledb\_array-method (extended), [29](#)
- extended<-, [30](#)
- extended<-, tiledb\_array-method (extended<-), [30](#)
- fetchBatched, [30](#)
- filter\_list (generics), [37](#)
- filter\_list, tiledb\_array\_schema-method, [31](#)
- filter\_list, tiledb\_attr-method, [31](#)
- filter\_list, tiledb\_dim-method, [32](#)
- filter\_list<-, tiledb\_attr-method, [32](#)
- filter\_list<-, tiledb\_dim-method, [33](#)
- filter\_list<- (generics), [37](#)
- fromDataFrame, [33](#)
- fromMatrix, [35](#)
- fromSparseMatrix, [36](#)
- generics, [37](#)
- get\_allocation\_size\_preference (save\_allocation\_size\_preference), [56](#)
- get\_return\_as\_preference (save\_return\_as\_preference), [57](#)
- has\_attribute, [38](#)
- is.anonymous, [39](#)
- is.anonymous.tiledb\_dim, [40](#)
- is.integral (generics), [37](#)
- is.integral, tiledb\_domain-method, [40](#)
- is.sparse (generics), [37](#)
- is.sparse, tiledb\_array\_schema-method, [41](#)
- limitTileDBCores, [41](#)
- load\_allocation\_size\_preference (save\_allocation\_size\_preference), [56](#)
- load\_return\_as\_preference (save\_return\_as\_preference), [57](#)

- max\_chunk\_size, 42
- max\_chunk\_size, tiledb\_filter\_list-method  
(max\_chunk\_size), 42
- name (generics), 37
- name, tiledb\_attr-method, 43
- name, tiledb\_dim-method, 44
- nfilters (generics), 37
- nfilters, tiledb\_filter\_list-method, 44
- parse\_query\_condition, 45
- print.tiledb\_metadata, 46
- query\_condition, 47
- query\_condition, tiledb\_array-method  
(query\_condition), 47
- query\_condition<-, 47
- query\_condition<-, tiledb\_array-method  
(query\_condition<-), 47
- query\_layout, 48
- query\_layout, tiledb\_array-method  
(query\_layout), 48
- query\_layout<-, 48
- query\_layout<-, tiledb\_array-method  
(query\_layout<-), 48
- query\_statistics, 49
- query\_statistics, tiledb\_array-method  
(query\_statistics), 49
- query\_statistics<-, 49
- query\_statistics<-, tiledb\_array-method  
(query\_statistics<-), 49
- r\_to\_tiledb\_type, 55
- raw\_dump (generics), 37
- raw\_dump, tiledb\_array\_schema-method,  
50
- raw\_dump, tiledb\_attr-method, 50
- raw\_dump, tiledb\_domain-method, 51
- return.array, 51
- return.array, tiledb\_array-method  
(return.array), 51
- return.array<-, 52
- return.array<-, tiledb\_array-method  
(return.array<-), 52
- return.data.frame (generics), 37
- return.data.frame, tiledb\_array-method,  
52
- return.data.frame<-, tiledb\_array-method,  
53
- return.data.frame<- (generics), 37
- return.matrix, 53
- return.matrix, tiledb\_array-method  
(return.matrix), 53
- return.matrix<-, 54
- return.matrix<-, tiledb\_array-method  
(return.matrix<-), 54
- return\_as, 54
- return\_as, tiledb\_array-method  
(return\_as), 54
- return\_as<-, 55
- return\_as<-, tiledb\_array-method  
(return\_as<-), 55
- save\_allocation\_size\_preference, 56
- save\_return\_as\_preference, 57
- schema (generics), 37
- schema, character-method, 58
- schema, tiledb\_array-method, 58
- schema\_check, 59
- schema\_check, tiledb\_array\_schema-method  
(schema\_check), 59
- selected\_points, 59
- selected\_points, tiledb\_array-method  
(selected\_points), 59
- selected\_points<-, 60
- selected\_points<-, tiledb\_array-method  
(selected\_points<-), 60
- selected\_ranges, 60
- selected\_ranges, tiledb\_array-method  
(selected\_ranges), 60
- selected\_ranges<-, 61
- selected\_ranges<-, tiledb\_array-method  
(selected\_ranges<-), 61
- set\_allocation\_size\_preference  
(save\_allocation\_size\_preference),  
56
- set\_max\_chunk\_size, 62
- set\_max\_chunk\_size, tiledb\_filter\_list, numeric-method  
(set\_max\_chunk\_size), 62
- set\_return\_as\_preference  
(save\_return\_as\_preference), 57
- show, tiledb\_array-method, 62
- show, tiledb\_array\_schema-method, 63
- show, tiledb\_attr-method, 63
- show, tiledb\_config-method, 64
- show, tiledb\_dim-method, 64
- show, tiledb\_domain-method, 65
- show, tiledb\_filter-method, 65

- show, tiledb\_filter\_list-method, 66
- show, tiledb\_group-method, 66
- statusBatched, 67
- strings\_as\_factors, 67
- strings\_as\_factors, tiledb\_array-method  
(strings\_as\_factors), 67
- strings\_as\_factors<-, 68
- strings\_as\_factors<-, tiledb\_array-method  
(strings\_as\_factors<-), 68
  
- tdb\_collect (generics), 37
- tdb\_collect, tiledb\_array-method, 68
- tdb\_filter (generics), 37
- tdb\_filter, tiledb\_array-method, 69
- tdb\_select (generics), 37
- tdb\_select, tiledb\_array-method, 69
- tile (generics), 37
- tile, tiledb\_dim-method, 70
- tile\_order (generics), 37
- tile\_order, tiledb\_array\_schema-method,  
190
- tiledb\_array, 70
- tiledb\_array-class, 73
- tiledb\_array\_apply\_aggregate, 74
- tiledb\_array\_close, 74
- tiledb\_array\_create, 75
- tiledb\_array\_delete\_fragments, 75
- tiledb\_array\_get\_non\_empty\_domain\_from\_index,  
76
- tiledb\_array\_get\_non\_empty\_domain\_from\_name,  
77
- tiledb\_array\_has\_enumeration, 77
- tiledb\_array\_is\_heterogeneous, 78
- tiledb\_array\_is\_homogeneous, 78
- tiledb\_array\_is\_open, 79
- tiledb\_array\_open, 79
- tiledb\_array\_open\_at, 80
- tiledb\_array\_schema, 80
- tiledb\_array\_schema-class, 81
- tiledb\_array\_schema\_check  
(schema\_check), 59
- tiledb\_array\_schema\_evolution, 82
- tiledb\_array\_schema\_evolution-class,  
82
- tiledb\_array\_schema\_evolution\_add\_attribute,  
83
- tiledb\_array\_schema\_evolution\_add\_enumeration,  
83
- tiledb\_array\_schema\_evolution\_add\_enumeration\_empty,  
84
- tiledb\_array\_schema\_evolution\_array\_evolve,  
84
- tiledb\_array\_schema\_evolution\_drop\_attribute,  
85
- tiledb\_array\_schema\_evolution\_drop\_enumeration,  
85
- tiledb\_array\_schema\_evolution\_extend\_enumeration,  
86
- tiledb\_array\_schema\_get\_allows\_dups  
(allows\_dups), 9
- tiledb\_array\_schema\_get\_capacity  
(capacity), 16
- tiledb\_array\_schema\_set\_allows\_dups  
(allows\_dups<-), 10
- tiledb\_array\_schema\_set\_capacity  
(capacity<-), 17
- tiledb\_array\_schema\_set\_coords\_filter\_list,  
87
- tiledb\_array\_schema\_set\_enumeration\_empty,  
87
- tiledb\_array\_schema\_set\_offsets\_filter\_list,  
88
- tiledb\_array\_schema\_set\_validity\_filter\_list,  
89
- tiledb\_array\_schema\_version, 89
- tiledb\_array\_upgrade\_version, 90
- tiledb\_arrow\_array\_del  
(tiledb\_arrow\_array\_ptr), 90
- tiledb\_arrow\_array\_ptr, 90
- tiledb\_arrow\_schema\_del  
(tiledb\_arrow\_array\_ptr), 90
- tiledb\_arrow\_schema\_ptr  
(tiledb\_arrow\_array\_ptr), 90
- tiledb\_attr, 91
- tiledb\_attr-class, 92
- tiledb\_attribute\_get\_cell\_size, 92
- tiledb\_attribute\_get\_cell\_val\_num  
(cell\_val\_num), 18
- tiledb\_attribute\_get\_enumeration, 92
- tiledb\_attribute\_get\_enumeration\_ptr  
(tiledb\_attribute\_get\_enumeration),  
92
- tiledb\_attribute\_get\_fill\_value, 93
- tiledb\_attribute\_get\_nullable, 93
- tiledb\_attribute\_has\_enumeration, 94
- tiledb\_attribute\_is\_ordered\_enumeration\_ptr,

- 94
- tiledb\_attribute\_is\_variable\_sized, 95
- tiledb\_attribute\_set\_cell\_val\_num  
(cell\_val\_num<-), 19
- tiledb\_attribute\_set\_enumeration\_name,  
95
- tiledb\_attribute\_set\_fill\_value, 96
- tiledb\_attribute\_set\_nullable, 96
- tiledb\_config, 97
- tiledb\_config-class, 97
- tiledb\_config\_as\_built\_json, 98
- tiledb\_config\_as\_built\_show, 98
- tiledb\_config\_load, 99
- tiledb\_config\_save, 99
- tiledb\_config\_unset, 100
- tiledb\_ctx, 100
- tiledb\_ctx-class, 101
- tiledb\_ctx\_set\_default\_tags, 101
- tiledb\_ctx\_set\_tag, 102
- tiledb\_ctx\_stats, 102
- tiledb\_datatype\_R\_type, 103
- tiledb\_delete\_metadata, 103
- tiledb\_dense (tiledb\_array), 70
- tiledb\_dim, 104
- tiledb\_dim-class, 105
- tiledb\_dim\_get\_cell\_val\_num  
(cell\_val\_num, tiledb\_dim-method),  
18
- tiledb\_domain, 105
- tiledb\_domain-class, 105
- tiledb\_domain\_get\_dimension\_from\_index,  
106
- tiledb\_domain\_get\_dimension\_from\_name,  
106
- tiledb\_domain\_has\_dimension, 107
- tiledb\_error\_message, 107
- tiledb\_filestore\_buffer\_export, 108
- tiledb\_filestore\_buffer\_import, 108
- tiledb\_filestore\_schema\_create, 109
- tiledb\_filestore\_size, 110
- tiledb\_filestore\_uri\_export, 110
- tiledb\_filestore\_uri\_import, 111
- tiledb\_filter, 111
- tiledb\_filter-class, 112
- tiledb\_filter\_get\_option, 113
- tiledb\_filter\_list, 113
- tiledb\_filter\_list-class, 114
- tiledb\_filter\_list\_get\_max\_chunk\_size  
(max\_chunk\_size), 42
- tiledb\_filter\_list\_set\_max\_chunk\_size  
(set\_max\_chunk\_size), 62
- tiledb\_filter\_set\_option, 114
- tiledb\_filter\_type, 115
- tiledb\_fragment\_info, 115
- tiledb\_fragment\_info-class, 116
- tiledb\_fragment\_info\_dense, 116
- tiledb\_fragment\_info\_dump, 117
- tiledb\_fragment\_info\_get\_cell\_num, 117
- tiledb\_fragment\_info\_get\_non\_empty\_domain\_index,  
118
- tiledb\_fragment\_info\_get\_non\_empty\_domain\_name,  
118
- tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_index,  
119
- tiledb\_fragment\_info\_get\_non\_empty\_domain\_var\_name,  
119
- tiledb\_fragment\_info\_get\_num, 120
- tiledb\_fragment\_info\_get\_size, 120
- tiledb\_fragment\_info\_get\_timestamp\_range,  
121
- tiledb\_fragment\_info\_get\_to\_vacuum\_num,  
121
- tiledb\_fragment\_info\_get\_to\_vacuum\_uri,  
122
- tiledb\_fragment\_info\_get\_unconsolidated\_metadata\_num,  
122
- tiledb\_fragment\_info\_get\_version, 123
- tiledb\_fragment\_info\_has\_consolidated\_metadata,  
123
- tiledb\_fragment\_info\_sparse, 124
- tiledb\_fragment\_info\_uri, 124
- tiledb\_get\_all\_metadata, 125
- tiledb\_get\_context, 125
- tiledb\_get\_metadata, 126
- tiledb\_get\_query\_status, 126
- tiledb\_get\_vfs, 127
- tiledb\_group, 127
- tiledb\_group-class, 128
- tiledb\_group\_add\_member, 128
- tiledb\_group\_close, 129
- tiledb\_group\_create, 129
- tiledb\_group\_delete\_metadata, 130
- tiledb\_group\_get\_all\_metadata, 130
- tiledb\_group\_get\_config, 131
- tiledb\_group\_get\_metadata, 131
- tiledb\_group\_get\_metadata\_from\_index,

- 132
- tiledb\_group\_has\_metadata, 132
- tiledb\_group\_is\_open, 133
- tiledb\_group\_is\_relative, 133
- tiledb\_group\_member, 134
- tiledb\_group\_member\_count, 134
- tiledb\_group\_member\_dump, 135
- tiledb\_group\_metadata\_num, 135
- tiledb\_group\_open, 136
- tiledb\_group\_put\_metadata, 136
- tiledb\_group\_query\_type, 137
- tiledb\_group\_remove\_member, 137
- tiledb\_group\_set\_config, 138
- tiledb\_group\_uri, 138
- tiledb\_has\_metadata, 139
- tiledb\_is\_supported\_fs, 139
- tiledb\_ndim (generics), 37
- tiledb\_ndim, tiledb\_array\_schema-method, 140
- tiledb\_ndim, tiledb\_dim-method, 141
- tiledb\_ndim, tiledb\_domain-method, 141
- tiledb\_num\_metadata, 142
- tiledb\_object\_ls, 142
- tiledb\_object\_mv, 143
- tiledb\_object\_rm, 143
- tiledb\_object\_type, 144
- tiledb\_object\_walk, 144
- tiledb\_put\_metadata, 145
- tiledb\_query, 145
- tiledb\_query-class, 146
- tiledb\_query\_add\_range, 146
- tiledb\_query\_add\_range\_with\_type, 147
- tiledb\_query\_alloc\_buffer\_ptr\_char, 147
- tiledb\_query\_apply\_aggregate, 148
- tiledb\_query\_buffer\_alloc\_ptr, 149
- tiledb\_query\_condition, 149
- tiledb\_query\_condition-class, 150
- tiledb\_query\_condition\_combine, 150
- tiledb\_query\_condition\_create, 151
- tiledb\_query\_condition\_init, 151
- tiledb\_query\_condition\_set\_use\_enumeration, 152
- tiledb\_query\_create\_buffer\_ptr, 153
- tiledb\_query\_create\_buffer\_ptr\_char, 153
- tiledb\_query\_ctx, 154
- tiledb\_query\_export\_buffer, 154
- tiledb\_query\_finalize, 155
- tiledb\_query\_get\_buffer\_char, 155
- tiledb\_query\_get\_buffer\_ptr, 156
- tiledb\_query\_get\_est\_result\_size, 156
- tiledb\_query\_get\_est\_result\_size\_var, 157
- tiledb\_query\_get\_fragment\_num, 157
- tiledb\_query\_get\_fragment\_timestamp\_range, 158
- tiledb\_query\_get\_fragment\_uri, 158
- tiledb\_query\_get\_layout, 159
- tiledb\_query\_get\_range, 159
- tiledb\_query\_get\_range\_num, 160
- tiledb\_query\_get\_range\_var, 160
- tiledb\_query\_import\_buffer, 161
- tiledb\_query\_result\_buffer\_elements, 161
- tiledb\_query\_result\_buffer\_elements\_vec, 162
- tiledb\_query\_set\_buffer, 163
- tiledb\_query\_set\_buffer\_ptr, 163
- tiledb\_query\_set\_buffer\_ptr\_char, 164
- tiledb\_query\_set\_condition, 164
- tiledb\_query\_set\_layout, 165
- tiledb\_query\_set\_subarray, 165
- tiledb\_query\_stats, 166
- tiledb\_query\_status, 166
- tiledb\_query\_submit, 167
- tiledb\_query\_submit\_async, 167
- tiledb\_query\_type, 168
- tiledb\_schema\_get\_dim\_attr\_status, 168
- tiledb\_schema\_get\_enumeration\_status, 169
- tiledb\_schema\_get\_names, 169
- tiledb\_schema\_get\_types, 170
- tiledb\_schema\_object, 170
- tiledb\_set\_context, 171
- tiledb\_set\_vfs, 171
- tiledb\_sparse (tiledb\_array), 70
- tiledb\_stats\_disable, 172
- tiledb\_stats\_dump, 172
- tiledb\_stats\_enable, 172
- tiledb\_stats\_print, 173
- tiledb\_stats\_raw\_dump, 173
- tiledb\_stats\_raw\_get, 173
- tiledb\_stats\_raw\_print, 174
- tiledb\_stats\_reset, 174
- tiledb\_subarray, 174

- tiledb\_subarray-class, 175
- tiledb\_subarray\_to\_query, 175
- tiledb\_version, 176
- tiledb\_vfs, 176
- tiledb\_vfs-class, 177
- tiledb\_vfs\_close, 177
- tiledb\_vfs\_copy\_file, 178
- tiledb\_vfs\_create\_bucket, 178
- tiledb\_vfs\_create\_dir, 179
- tiledb\_vfs\_dir\_size, 179
- tiledb\_vfs\_empty\_bucket, 180
- tiledb\_vfs\_file\_size, 180
- tiledb\_vfs\_is\_bucket, 181
- tiledb\_vfs\_is\_dir, 181
- tiledb\_vfs\_is\_empty\_bucket, 182
- tiledb\_vfs\_is\_file, 183
- tiledb\_vfs\_ls, 183
- tiledb\_vfs\_ls\_recursive, 184
- tiledb\_vfs\_move\_dir, 184
- tiledb\_vfs\_move\_file, 185
- tiledb\_vfs\_open, 185
- tiledb\_vfs\_read, 186
- tiledb\_vfs\_remove\_bucket, 186
- tiledb\_vfs\_remove\_dir, 187
- tiledb\_vfs\_remove\_file, 187
- tiledb\_vfs\_serialize, 188
- tiledb\_vfs\_sync, 188
- tiledb\_vfs\_touch, 189
- tiledb\_vfs\_unserialize, 189
- tiledb\_vfs\_write, 190
- toMatrix (fromMatrix), 35
- toSparseMatrix (fromSparseMatrix), 36