

Package ‘pizzarr’

August 22, 2026

Type Package

Title Slice into 'Zarr' Arrays

Version 0.2.1

Description An implementation of chunked, compressed,
N-dimensional arrays for R. 'Zarr' spec V2 (2024)
<[doi:10.5281/zenodo.11320255](https://doi.org/10.5281/zenodo.11320255)>.

Language en-US

License MIT + file LICENSE

BugReports <https://github.com/zarr-developers/pizzarr/issues>

URL <https://zarr.dev/pizzarr/>,
<https://github.com/zarr-developers/pizzarr>

Depends R (>= 4.2)

Imports jsonlite, stats, R6, qs2, stringr, memoise, utils

Encoding UTF-8

RoxygenNote 7.3.3

VignetteBuilder knitr

Suggests testthat, knitr, covr, bslib, pkgdown, rmarkdown, crul,
blosc, vcr (>= 0.6.0), bench, bit64, withr

Config/testthat/parallel false

Config/testthat/edition 3

NeedsCompilation no

Author David Blodgett [cre, aut] (ORCID:
<<https://orcid.org/0000-0001-9489-1710>>),
Mark Keller [aut] (ORCID: <<https://orcid.org/0000-0003-3003-874X>>),
Artür Manukyan [aut] (ORCID: <<https://orcid.org/0000-0002-0441-9517>>),
zarr-developers [cph]

Maintainer David Blodgett <dblodgett@usgs.gov>

Repository CRAN

Date/Publication 2026-08-22 17:10:02 UTC

Contents

as_scalar	2
Attributes	3
BloscCodec	6
Bz2Codec	8
Codec	9
Dtype	11
GzipCodec	12
HttpStore	14
int	17
is_key_error	17
is_scalar	18
is_slice	18
LzmaCodec	19
NestedArray	20
pizzarr_compiled_features	22
pizzarr_config	23
pizzarr_option_defaults	24
pizzarr_sample	24
pizzarr_upgrade	25
slice	25
VLenUtf8Codec	26
zarrs_compiled_features	28
zarr_create	28
zarr_create_array	30
zarr_create_empty	30
zarr_create_group	31
zarr_create_zeros	32
zarr_open	32
zarr_open_array	33
zarr_open_group	35
zarr_save_array	36
zarr_volcano	36
zb_int	37
zb_slice	37
ZlibCodec	38

Index	40
--------------	-----------

as_scalar	<i>Convert a value to a scalar to opt-out of R default vector casting behavior. This uses the <code>jsonlite::unbox</code> function to "tag" the value as a scalar.</i>
-----------	---

Description

Convert a value to a scalar to opt-out of R default vector casting behavior. This uses the `jsonlite::unbox` function to "tag" the value as a scalar.

Usage

`as_scalar(obj)`

Arguments

`obj` The value to convert.

Value

The value wrapped as a scalar.

Attributes	<i>Attributes Class</i>
------------	-------------------------

Description

Class providing access to user attributes on an array or group.

Format

[R6::R6Class](#)

Details

The Zarr Attributes class.

Public fields

- `store` Attributes store, already initialized.
- `key` The key under which the attributes will be stored.
- `read_only` If True, attributes cannot be modified.
- `cache` If True (default), attributes will be cached locally.
- `synchronizer` Only necessary if attributes may be modified from multiple threads or processes.

Methods

Public methods:

- `Attributes$new()`
- `Attributes$to_list()`
- `Attributes$refresh()`
- `Attributes$contains()`
- `Attributes$get_item()`
- `Attributes$set_item()`
- `Attributes$del_item()`
- `Attributes$set_cached_v3_attrs()`
- `Attributes$clone()`

Method `new()`: Create a new Attributes instance.

Usage:

```
Attributes$new(
  store,
  key = NA,
  read_only = FALSE,
  cache = TRUE,
  synchronizer = NA,
  zarr_format = NULL
)
```

Arguments:

`store` ([Store](#))

Attributes store, already initialized.

`key` (`character(1)`)

Key to use for attributes (`.zattrs` is default).

`read_only` (`logical(1)`)

Whether the attributes are read-only.

`cache` (`logical(1)`)

Whether to cache attributes.

`synchronizer` (ANY or NA)

Synchronizer object.

`zarr_format` (`integer(1)` or NULL)

Zarr format version: 2L for V2 (`.zattrs`), 3L for V3 (`zarr.json`).

Returns: An Attributes instance.

Method `to_list()`: convert attributes to list

Usage:

```
Attributes$to_list()
```

Returns: `list()`.

Method `refresh()`: refresh attributes

Usage:

Attributes\$refresh()

Returns: NULL (called for side effects).

Method contains(): check if object contains item

Usage:

Attributes\$contains(x)

Arguments:

x Object to test.

Returns: logical(1).

Method get_item(): get attribute

Usage:

Attributes\$get_item(item)

Arguments:

item Character attribute name.

Returns: The attribute value.

Method set_item(): set attribute

Usage:

Attributes\$set_item(item, value)

Arguments:

item Character attribute name.

value Value to add or update.

Returns: NULL (called for side effects).

Method del_item(): delete attribute

Usage:

Attributes\$del_item(item)

Arguments:

item Character attribute name.

Returns: NULL (called for side effects).

Method set_cached_v3_attrs(): Set cached attributes from V3 embedded metadata. In V3, attributes are part of zarr.json rather than a separate .zattrs file. This method pre-populates the cache so the normal .zattrs read path is skipped.

Usage:

Attributes\$set_cached_v3_attrs(attrs_list)

Arguments:

attrs_list A named list of attributes from V3 zarr.json.

Returns: NULL (modifies cache in place).

Method clone(): The objects of this class are cloneable with this method.

Usage:

Attributes\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

BloscCodec

*BloscCodec Class***Description**

Class representing a Blosc compressor

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Blosc compressor for Zarr

Super class

[pizzarr::Codec](#) -> BloscCodec

Public fields

`cname` (character(1))
The compression algorithm to use.

`clevel` (integer(1))
The compression level.

`shuffle` (logical(1) | integer(1))
The shuffle filter to use.

`blocksize` (integer(1) | NA)
The block size.

Methods**Public methods:**

- [BloscCodec\\$new\(\)](#)
- [BloscCodec\\$encode\(\)](#)
- [BloscCodec\\$decode\(\)](#)
- [BloscCodec\\$get_config\(\)](#)
- [BloscCodec\\$clone\(\)](#)

Method `new()`: Create a new Blosc compressor.

Usage:

```
BloscCodec$new(cname = "lz4", clevel = 5, shuffle = TRUE, blocksize = NA, ...)
```

Arguments:

`cname` (character(1))
The compression algorithm to use.

clevel (integer(1))
 The compression level.
shuffle (logical(1) | integer(1))
 The shuffle filter to use.
blocksize (integer(1) | NA)
 The block size.
... Not used.
Returns: A new BloscCodec object.

Method encode(): Compress data.

Usage:
BloscCodec\$encode(buf, zarr_arr)
Arguments:
buf (raw())
 The un-compressed data.
zarr_arr ([ZarrArray](#))
 The ZarrArray instance.
Returns: Compressed data.

Method decode(): Decompress data.

Usage:
BloscCodec\$decode(buf, zarr_arr)
Arguments:
buf (raw())
 The compressed data.
zarr_arr ([ZarrArray](#))
 The ZarrArray instance.
Returns: Un-compressed data.

Method get_config(): Get codec configuration as a list.

Usage:
BloscCodec\$get_config()
Returns: A named list.

Method clone(): The objects of this class are cloneable with this method.

Usage:
BloscCodec\$clone(deep = FALSE)
Arguments:
deep Whether to make a deep clone.

See Also

Other Codec classes: [Bz2Codec](#), [Codec](#), [GzipCodec](#), [Lz4Codec](#), [LzmaCodec](#), [VLenUtf8Codec](#), [ZlibCodec](#), [ZstdCodec](#)

Bz2Codec

Bz2Codec Class

Description

Class representing a bz2 compressor

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Bz2 compressor for Zarr

Super class

[pizzarr::Codec](#) -> Bz2Codec

Public fields

level The compression level.

Methods

Public methods:

- [Bz2Codec\\$new\(\)](#)
- [Bz2Codec\\$encode\(\)](#)
- [Bz2Codec\\$decode\(\)](#)
- [Bz2Codec\\$get_config\(\)](#)
- [Bz2Codec\\$clone\(\)](#)

Method [new\(\)](#): Create a new Bz2 compressor.

Usage:

[Bz2Codec\\$new](#)(level = 6, ...)

Arguments:

level The compression level, between 1 and 22.

... Not used.

Returns: A new Bz2Codec object.

Method [encode\(\)](#): Compress data.

Usage:

[Bz2Codec\\$encode](#)(buf, zarr_arr)

Arguments:

buf (raw())
The un-compressed data.
zarr_arr ([ZarrArray](#))
The ZarrArray instance.

Returns: Compressed data.

Method decode(): Decompress data.

Usage:
Bz2Codec\$decode(buf, zarr_arr)

Arguments:
buf (raw())
The compressed data.
zarr_arr ([ZarrArray](#))
The ZarrArray instance.

Returns: Un-compressed data.

Method get_config(): Get codec configuration as a list.

Usage:
Bz2Codec\$get_config()

Returns: A named list.

Method clone(): The objects of this class are cloneable with this method.

Usage:
Bz2Codec\$clone(deep = FALSE)

Arguments:
deep Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Codec](#), [GzipCodec](#), [Lz4Codec](#), [LzmaCodec](#), [VLenUtf8Codec](#), [ZlibCodec](#), [ZstdCodec](#)

Codec	<i>Codec Class</i>
-------	--------------------

Description

Abstract class representing a compressor.

Format

[R6::R6Class](#)

Details

Abstract compressor for Zarr

Methods

Public methods:

- [Codec\\$encode\(\)](#)
- [Codec\\$decode\(\)](#)
- [Codec\\$get_config\(\)](#)
- [Codec\\$clone\(\)](#)

Method `encode()`: Compress data.

Usage:

`Codec$encode(buf, zarr_arr)`

Arguments:

`buf` ([raw\(\)](#))

The un-compressed data.

`zarr_arr` ([ZarrArray](#))

The ZarrArray instance.

Returns: Compressed data.

Method `decode()`: Decompress data.

Usage:

`Codec$decode(buf, zarr_arr)`

Arguments:

`buf` ([raw\(\)](#))

The compressed data.

`zarr_arr` ([ZarrArray](#))

The ZarrArray instance.

Returns: Un-compressed data.

Method `get_config()`: Get codec configuration as a list.

Usage:

`Codec$get_config()`

Returns: A named list.

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

Usage:

`Codec$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Bz2Codec](#), [GzipCodec](#), [Lz4Codec](#), [LzmaCodec](#), [VLenUtf8Codec](#), [ZlibCodec](#), [ZstdCodec](#)

Dtype

*Dtype Class***Description**

A data type object (an instance of Dtype class) describes how the bytes in the fixed-size block of memory corresponding to an array item should be interpreted.

Format

[R6::R6Class](#)

Details

The Zarr Dtype class.

Public fields

`dtype` The original dtype string, like "<f4".
`byte_order` The byte order of the dtype, either "little", "big", or "nr".
`basic_type` The basic type of the dtype, like "f".
`num_bytes` The number of bytes of the dtype.
`num_items` The number of items of the dtype.
`is_signed` Whether the dtype is signed. Logical/boolean.
`is_structured` Whether the dtype is structured. Logical/boolean.
`is_object` Whether the dtype is an object. Logical/boolean.
`object_codec` The object codec instance.

Methods**Public methods:**

- [Dtype\\$new\(\)](#)
- [Dtype\\$get_asrtype\(\)](#)
- [Dtype\\$get_rtype\(\)](#)
- [Dtype\\$get_typed_array_ctr\(\)](#)
- [Dtype\\$clone\(\)](#)

Method `new()`: Create a new Dtype instance.

Usage:

```
Dtype$new(dtype, object_codec = NA)
```

Arguments:

`dtype` The original dtype string, like "<f4".
`object_codec` The object codec instance.

Returns: A Dtype instance.

Method `get_asrtype()`: Get the R coercion function name for this dtype.

Usage:

`Dtype$get_asrtype()`

Returns: Character string (e.g., "as.double").

Method `get_rtype()`: Get the R base type for this dtype.

Usage:

`Dtype$get_rtype()`

Returns: An R prototype value (e.g., `integer()`, `double()`, or `bit64::integer64()`).

Method `get_typed_array_ctr()`: Get a constructor function for typed arrays of this dtype.

Usage:

`Dtype$get_typed_array_ctr()`

Returns: A function that takes `dim` and returns an array.

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

Usage:

`Dtype$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

References

<https://numpy.org/doc/stable/reference/arrays.dtypes.html>

GzipCodec

GzipCodec Class

Description

Class representing a gzip compressor.

Gzip encoding uses temporary files because R's `memCompress()` produces zlib framing rather than gzip framing. This makes GzipCodec slower than [ZstdCodec](#) for writes. Prefer ZstdCodec when performance matters.

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Gzip compressor for Zarr

Super class

`pizzarr::Codec` -> GzipCodec

Public fields

level The compression level.

Methods**Public methods:**

- `GzipCodec$new()`
- `GzipCodec$encode()`
- `GzipCodec$decode()`
- `GzipCodec$get_config()`
- `GzipCodec$clone()`

Method `new()`: Create a new Gzip compressor.

Usage:

`GzipCodec$new(level = 6, ...)`

Arguments:

level The compression level, between 1 and 22.

... Not used.

Returns: A new GzipCodec object.

Method `encode()`: Compress data.

Usage:

`GzipCodec$encode(buf, zarr_arr)`

Arguments:

buf (`raw()`)

The un-compressed data.

zarr_arr (`ZarrArray`)

The ZarrArray instance.

Returns: Compressed data.

Method `decode()`: Decompress data.

Usage:

`GzipCodec$decode(buf, zarr_arr)`

Arguments:

buf (`raw()`)

The compressed data.

zarr_arr (`ZarrArray`)

The ZarrArray instance.

Returns: Un-compressed data.

Method `get_config()`: Get codec configuration as a list.

Usage:

```
GzipCodec$get_config()
```

Returns: A named list.

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

Usage:

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

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Bz2Codec](#), [Codec](#), [Lz4Codec](#), [LzmaCodec](#), [VLenUtf8Codec](#), [ZlibCodec](#), [ZstdCodec](#)

HttpStore

HttpStore Class

Description

Store class that uses HTTP requests. Read-only. Depends on the `curl` package.

Format

[R6::R6Class](#) inheriting from [Store](#).

Details

This is also the way to read an **S3-compatible bucket over plain HTTPS**. Most public object stores — MinIO, Ceph, Open Storage Network pods, Cloudflare R2, and Google Cloud Storage — expose their objects at a URL, so an `s3://bucket/key` address with endpoint `https://host` becomes:

```
https://host/bucket/key
```

Unlike [S3Store](#) and [GcsStore](#), `HttpStore` works on both distribution tiers: it falls back to the R-native `curl` code path when the `zarrs` backend is not compiled in, and uses parallel chunk decode when it is. It needs no credentials and no environment configuration.

When the store publishes consolidated metadata (a `.zmetadata` key), `listdir()` reports its members; without it, HTTP stores cannot be listed and you must address arrays by name.

Super class

[pizzarr::Store](#) -> `HttpStore`

Methods**Public methods:**

- `HttpStore$new()`
- `HttpStore$get_item()`
- `HttpStore$contains_item()`
- `HttpStore$listdir()`
- `HttpStore$get_cache_time_seconds()`
- `HttpStore$set_cache_time_seconds()`
- `HttpStore$get_store_identifier()`
- `HttpStore$print()`
- `HttpStore$clone()`

Method `new()`: Create a `HttpStore` object

Usage:

```
HttpStore$new(url, options = NA, headers = NA)
```

Arguments:

`url` (`character(1)`)

URL of the store.

`options` (`list()` or `NA`)

Options passed to `curl`.

`headers` (`list()` or `NA`)

Headers passed to `curl`.

Returns: A new `HttpStore` object.

Method `get_item()`: Get an item from the store.

Usage:

```
HttpStore$get_item(item)
```

Arguments:

`item` The item key.

Returns: The item data in a vector of type `raw`.

Method `contains_item()`: Determine whether the store contains an item.

Usage:

```
HttpStore$contains_item(item)
```

Arguments:

`item` The item key.

Returns: A boolean value.

Method `listdir()`: Fetches `.zmetadata` from the store evaluates its names

Usage:

```
HttpStore$listdir(path = NA)
```

Arguments:

path character path to list within, or NA for the store root.

Returns: Character vector of unique keys that do not start with a ..

Method `get_cache_time_seconds()`: Get cache time of http requests.

Usage:

```
HttpStore$get_cache_time_seconds()
```

Returns: numeric(1).

Method `set_cache_time_seconds()`: Set cache time of http requests.

Usage:

```
HttpStore$set_cache_time_seconds(seconds)
```

Arguments:

seconds Number of seconds until cache is invalid – 0 for no cache.

Returns: NULL (called for side effects).

Method `get_store_identifier()`: Print a human-readable summary of the store.

Return the store URL for zarrs dispatch.

Usage:

```
HttpStore$get_store_identifier()
```

Returns: A character string.

Method `print()`:

Usage:

```
HttpStore$print(...)
```

Arguments:

... Ignored.

Returns: self (invisibly).

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

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

[S3Store](#) for s3:// URLs via the zarrs backend. `vignette("remote-stores")` for worked examples.

Other Store classes: [DirectoryStore](#), [GcsStore](#), [MemoryStore](#), [S3Store](#), [Store](#)

Examples

```
## Not run:
# USGS gridMET on an Open Storage Network pod, addressed over HTTPS.
store <- HttpStore$new("https://usgs.osn.mghpcc.org/mdmf/gdp/gridMET.zarr")
g <- zarr_open(store)
store$listdir()

## End(Not run)
```

int

Convenience function for the internal Int class constructor.

Description

Convenience function for the internal Int class constructor.

Usage

```
int(index, zero_based = FALSE)
```

Arguments

index	The integer index.
zero_based	The index of the dimension. By default, FALSE for R-like behavior.

Value

A Int instance with the specified parameters.

is_key_error

Check if an error is a KeyError.

Description

Check if an error is a KeyError.

Usage

```
is_key_error(e)
```

Arguments

e	The error to check.
---	---------------------

Value

TRUE if the error is a KeyError, FALSE otherwise.

is_scalar	<i>Check if a value is a scalar (i.e., a one-element vector that was converted with as_scalar).</i>
-----------	---

Description

Check if a value is a scalar (i.e., a one-element vector that was converted with as_scalar).

Usage

```
is_scalar(s)
```

Arguments

s	The value to check.
---	---------------------

Value

TRUE if the value is a scalar, FALSE otherwise.

is_slice	<i>Check if a value is a Slice instance.</i>
----------	--

Description

Check if a value is a Slice instance.

Usage

```
is_slice(s)
```

Arguments

s	The value to check.
---	---------------------

Value

TRUE if the value is a Slice instance, FALSE otherwise.

LzmaCodec

LzmaCodec Class

Description

Class representing a lzma compressor

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Lzma compressor for Zarr

Super class

[pizzarr::Codec](#) -> LzmaCodec

Public fields

level The compression level.

format The compression format.

Methods

Public methods:

- [LzmaCodec\\$new\(\)](#)
- [LzmaCodec\\$encode\(\)](#)
- [LzmaCodec\\$decode\(\)](#)
- [LzmaCodec\\$get_config\(\)](#)
- [LzmaCodec\\$clone\(\)](#)

Method [new\(\)](#): Create a new lzma compressor.

Usage:

```
LzmaCodec$new(level = 9, format = 1, ...)
```

Arguments:

level The compression level, between 1 and 22.

format (integer(1))

Only 1 is supported.

... Not used.

Returns: A new LzmaCodec object.

Method [encode\(\)](#): Compress data.

Usage:

```
LzmaCodec$encode(buf, zarr_arr)
```

Arguments:

buf (raw())

The un-compressed data.

zarr_arr ([ZarrArray](#))

The ZarrArray instance.

Returns: Compressed data.

Method decode(): Decompress data.

Usage:

```
LzmaCodec$decode(buf, zarr_arr)
```

Arguments:

buf (raw())

The compressed data.

zarr_arr ([ZarrArray](#))

The ZarrArray instance.

Returns: Un-compressed data.

Method get_config(): Get codec configuration as a list.

Usage:

```
LzmaCodec$get_config()
```

Returns: A named list.

Method clone(): The objects of this class are cloneable with this method.

Usage:

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

Arguments:

deep Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Bz2Codec](#), [Codec](#), [GzipCodec](#), [Lz4Codec](#), [VLenUtf8Codec](#), [ZlibCodec](#), [ZstdCodec](#)

NestedArray

NestedArray Class

Description

Represents a multi-dimensional array that can be accessed and subsetting via list of Slice instances.

Format

[R6::R6Class](#)

Details

The Zarr NestedArray class.

Public fields

`shape` The shape of the array.

`dtype` The Zarr dtype of the array, as a string like ">f8".

`dtype_obj` The Zarr dtype of the array, as a Dtype instance.

`data` The array contents as a base R array.

Methods

Public methods:

- [NestedArray\\$new\(\)](#)
- [NestedArray\\$get\(\)](#)
- [NestedArray\\$set\(\)](#)
- [NestedArray\\$flatten\(\)](#)
- [NestedArray\\$flatten_to_raw\(\)](#)
- [NestedArray\\$as.array\(\)](#)
- [NestedArray\\$clone\(\)](#)

Method `new()`: Create a new NestedArray instance.

Usage:

```
NestedArray$new(data, shape = NA, dtype = NA, order = NA)
```

Arguments:

`data` The data to initialize the array with. Either NULL, base R array, base R vector (numeric/logical), scalar, or raw vector.

`shape` The shape of the array.

`dtype` The Zarr dtype of the array, as a string like ">f8".

`order` The order of the array, either "C" or "F". Only used when data is a raw vector. Optional.

Returns: A NestedArray instance.

Method `get()`: Subset the array.

Usage:

```
NestedArray$get(selection)
```

Arguments:

`selection` A list of slices.

Returns: A new NestedArray (potentially a subset) representing the selection.

Method `set()`: Set a subset of the array.

Usage:

```
NestedArray$set(selection, value)
```

Arguments:

selection A list of slices.

value A NestedArray or a base R array.

Returns: NULL (called for side effects, modifies self\$data in place).

Method flatten(): Flatten the array contents.

Usage:

NestedArray\$flatten(order = NA)

Arguments:

order Either "C", "F", or NA.

Returns: The data as a flat vector.

Method flatten_to_raw(): Flatten the array contents and convert to a raw vector.

Usage:

NestedArray\$flatten_to_raw(order = NA)

Arguments:

order Either "C", "F", or NA.

Returns: The data as a flat raw() vector (or plain vector for object dtypes).

Method as.array(): Convert NestedArray to a base R array.

Usage:

NestedArray\$as.array()

Returns: array().

Method clone(): The objects of this class are cloneable with this method.

Usage:

NestedArray\$clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

pizzarr_compiled_features

List compiled zarrs features

Description

Returns the feature flags compiled into the zarrs backend, or character(0) with a message when the backend is absent.

Usage

```
pizzarr_compiled_features()
```

Value

Character vector of feature names (e.g. "filesystem", "gzip").

pizzarr_config	<i>Get or set pizzarr configuration</i>
----------------	---

Description

Controls parallelism and HTTP behaviour for the zarrs backend. Called with no arguments, returns the current settings as a named list. Called with arguments, sets the specified options and applies them to the Rust backend immediately.

Usage

```
pizzarr_config(
  nthreads = NULL,
  concurrent_target = NULL,
  http_batch_range_requests = NULL
)
```

Arguments

nthreads	Integer or NULL. Number of threads for the rayon thread pool. NULL uses all CPUs (the default). The pool can only be initialised once per R session; later changes require a restart. Use the PIZZARR_NTHREADS environment variable for reliable session-level control.
concurrent_target	Integer or NULL. Codec concurrency level — how many codec operations zarrs runs in parallel within a single read/write call. NULL uses the zarrs default (CPU count). Can be changed at any time.
http_batch_range_requests	Logical or NULL. Whether HTTP stores use multipart range requests (default TRUE). Set to FALSE for servers with incomplete multipart range support. Takes effect on the next zarr_open() or zarrs_get_subset() call that opens a new HTTP store; existing cached stores are not affected (use zarrs_close_store(url) to force re-creation).

Value

When called with no arguments, a named list of current settings. When called with arguments, the previous values (invisibly).

pizzarr_option_defaults

pizzarr_option_defaults

Description

- pizzarr.http_store_cache_time_seconds how long to cache web requests
- pizzarr.nthreads number of threads for parallel codec operations (NULL = all CPUs). Set-once: takes effect only before the first zarrs operation. Use env var PIZZARR_NTHREADS for reliable session-level control.
- pizzarr.concurrent_target codec concurrency level — how many codec operations zarrs runs in parallel within a single read/write call (NULL = zarrs default, typically CPU count). Can be changed at any time.
- pizzarr.http_batch_range_requests whether HTTP stores use multipart range requests (TRUE by default). Set to FALSE for servers with incomplete multipart range support. Takes effect on next store open.

Usage

pizzarr_option_defaults

Format

An object of class list of length 4.

pizzarr_sample

pizzarr demo data

Description

pizzarr demo data

Usage

```
pizzarr_sample(
  dataset = NULL,
  outdir = file.path(tools::R_user_dir("pizzarr"), "pizzarr_sample")
)
```

Arguments

dataset	character defining which demo dataset is desired, If NULL, all are returned
outdir	character directory path to store sample zarr stores

Details

For directory stores, unzips the store to a temporary directory and returns the resulting path.

Value

path to ready to use zarr store

Examples

```
sample_dir <- tools::R_user_dir("pizzarr")

clean <- !dir.exists(sample_dir)

zarr_samples <- pizzarr_sample(outdir = sample_dir)

#printing without system path for example
gsub(sample_dir, "...", zarr_samples, fixed = TRUE)

# clean up if you don't want to keep them for next time
if(clean) unlink(sample_dir, recursive = TRUE)
```

pizzarr_upgrade	<i>Upgrade to the zarrs backend</i>
-----------------	-------------------------------------

Description

Prints the command to install pizzarr from r-universe with the compiled zarrs backend, or reports that zarrs is already available.

Usage

```
pizzarr_upgrade()
```

slice	<i>Convenience function for the internal Slice R6 class constructor.</i>
-------	--

Description

Convenience function for the internal Slice R6 class constructor.

Usage

```
slice(start, stop = NA, step = NA, zero_based = FALSE)
```

Arguments

start	The start index.
stop	The stop index.
step	The step size. Negative values reverse the direction of the slice, matching Python/NumPy semantics (e.g., step = -1 iterates backward from start toward stop).
zero_based	The index of the dimension. By default, FALSE for R-like behavior.

Value

A Slice instance with the specified parameters.

Examples

```
g <- zarr_volcano()
v <- g$get_item("volcano")

# Reverse the first 5 columns of row 1 (zero-based: 5:0:-1)
v$get_orthogonal_selection(list(zb_slice(0, 1), zb_slice(5, 0, -1)))

# Full reverse of row 1 (zero-based: -1::-1)
v$get_orthogonal_selection(list(zb_slice(0, 1), zb_slice(-1, NA, -1)))
```

VLenUtf8Codec	<i>VLenUtf8Codec Class</i>
---------------	----------------------------

Description

Class representing a VLenUtf8 compressor

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Variable-length UTF-8 codec for Zarr

Super class

[pizzarr::Codec](#) -> VLenUtf8Codec

Methods

Public methods:

- [VLenUtf8Codec\\$encode\(\)](#)
- [VLenUtf8Codec\\$decode\(\)](#)
- [VLenUtf8Codec\\$get_config\(\)](#)
- [VLenUtf8Codec\\$clone\(\)](#)

Method `encode()`: Compress data.

Usage:

`VLenUtf8Codec$encode(buf, zarr_arr)`

Arguments:

`buf` (`character()`)

The un-compressed data (character vector).

`zarr_arr` ([ZarrArray](#))

The ZarrArray instance.

Returns: Compressed data.

Method `decode()`: Decompress data.

Usage:

`VLenUtf8Codec$decode(buf, zarr_arr)`

Arguments:

`buf` (`raw()`)

The compressed data.

`zarr_arr` ([ZarrArray](#))

The ZarrArray instance.

Returns: Un-compressed data.

Method `get_config()`: Get codec configuration as a list.

Usage:

`VLenUtf8Codec$get_config()`

Returns: A named list.

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

Usage:

`VLenUtf8Codec$clone(deep = FALSE)`

Arguments:

`deep` Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Bz2Codec](#), [Codec](#), [GzipCodec](#), [Lz4Codec](#), [LzmaCodec](#), [ZlibCodec](#), [ZstdCodec](#)

zarrs_compiled_features

Return compiled zarrs feature flags as a character vector.

Description

Called once at .onLoad to populate .pizzarr_env\$zarrs_available. The feature list is determined at compile time via cfg! checks. Also installs a no-op panic hook on first call.

Usage

```
zarrs_compiled_features()
```

zarr_create

Create an empty array

Description

Create an empty array

Usage

```
zarr_create(
  shape,
  chunks = TRUE,
  dtype = NA,
  compressor = NA,
  fill_value = NA,
  order = NA,
  store = NA,
  synchronizer = NA,
  overwrite = FALSE,
  path = NA,
  chunk_store = NA,
  filters = NA,
  cache_metadata = TRUE,
  cache_attrs = TRUE,
  read_only = FALSE,
  object_codec = NA,
  dimension_separator = NA,
  write_empty_chunks = TRUE,
  zarr_format = 2L,
  dimension_names = NULL
)
```

Arguments

shape	: int or tuple of ints Array shape.
chunks	: int or tuple of ints, optional Chunk shape. If True, will be guessed from shape and dtype. If False, will be set to shape, i.e., single chunk for the whole array. If an int, the chunk size in each dimension will be given by the value of chunks. Default is True.
dtype	: string or dtype, optional NumPy dtype.
compressor	: Codec, optional Primary compressor.
fill_value	: object Default value to use for uninitialized portions of the array.
order	: 'C', 'F', optional Memory layout to be used within each chunk.
store	: Store A mapping that supports string keys and bytes-like values.
synchronizer	: object, optional Array synchronizer.
overwrite	: bool, optional If True, erase all data in store prior to initialisation.
path	: string, bytes, optional Path under which array is stored.
chunk_store	: Store, optional Separate storage for chunks. If not provided, store will be used for storage of both chunks and metadata.
filters	: sequence, optional Sequence of filters to use to encode chunk data prior to compression.
cache_metadata	: bool, optional If True, array configuration metadata will be cached for the lifetime of the object. If False, array metadata will be reloaded prior to all data access and modification operations (may incur overhead depending on storage and data access pattern).
cache_attrs	: bool, optional If True (default), user attributes will be cached for attribute read operations. If False, user attributes are reloaded from the store prior to all attribute read operations.
read_only	: bool, optional True if array should be protected against modification.
object_codec	: Codec, optional A codec to encode object arrays, only needed if dtype=object.
dimension_separator	: '.', '/', optional Separator placed between the dimensions of a chunk.
write_empty_chunks	: bool, optional If True (default), all chunks will be stored regardless of their contents. If False, each chunk is compared to the array's fill value prior to storing. If a chunk is uniformly equal to the fill value, then that chunk is not be stored, and the store entry for that chunk's key is deleted. This setting enables sparser storage, as only chunks with non-fill-value data are stored, at the expense of overhead associated with checking the data of each chunk.
zarr_format	: int, optional Zarr format version. Use 2L (default) for Zarr V2 or 3L for Zarr V3.
dimension_names	: character vector, optional Named dimensions for V3 arrays. Length must equal length(shape). V3 only – passing a non-NULL value with zarr_format = 2L is an error. Default NULL.

Value

ZarrArray

<code>zarr_create_array</code>	<i>Create an array initialized with data.</i>
--------------------------------	---

Description

Create an array initialized with data.

Usage

`zarr_create_array(data, ...)`

Arguments

- `data` A base R `array()` or `pizzarr NestedArray` instance.
- `...` The params of `zarr_create()`

Value

ZarrArray

<code>zarr_create_empty</code>	<i>Create an array filled with NAs.</i>
--------------------------------	---

Description

Create an array filled with NAs.

Usage

`zarr_create_empty(shape, ...)`

Arguments

- `shape` : int or tuple of ints Array shape.
- `...` The params of `zarr_create()`

Value

ZarrArray

zarr_create_group	<i>Create a group.</i>
-------------------	------------------------

Description

Create a group.

Usage

```
zarr_create_group(
    store = NA,
    overwrite = FALSE,
    chunk_store = NA,
    cache_attrs = TRUE,
    synchronizer = NA,
    path = NA,
    zarr_format = 2L
)
```

Arguments

store	: Store A mapping that supports string keys and bytes-like values.
overwrite	: bool, optional If True, erase all data in store prior to initialisation.
chunk_store	: Store, optional Separate storage for chunks. If not provided, store will be used for storage of both chunks and metadata.
cache_attrs	: bool, optional If True (default), user attributes will be cached for attribute read operations. If False, user attributes are reloaded from the store prior to all attribute read operations.
synchronizer	: object, optional Array synchronizer.
path	: string, bytes, optional Path under which array is stored.
zarr_format	: int, optional Zarr format version. Use 2L (default) for Zarr V2 or 3L for Zarr V3.

Value

ZarrGroup

zarr_create_zeros	<i>Create an array filled with zeros.</i>
-------------------	---

Description

Create an array filled with zeros.

Usage

```
zarr_create_zeros(shape, ...)
```

Arguments

shape	: int or tuple of ints Array shape.
...	The params of zarr_create()

Value

ZarrArray

zarr_open	<i>Convenience function to open a group or array using file-mode-like semantics.</i>
-----------	--

Description

Convenience function to open a group or array using file-mode-like semantics.

Usage

```
zarr_open(store = NA, mode = NA, path = NA, ...)
```

Arguments

store	A Store object, or a character string that is coerced to one: an http:// or https:// URL becomes an HttpStore , an s3:// URL becomes an S3Store , a gs:// URL becomes a GcsStore , and anything else is treated as a filesystem path and becomes a DirectoryStore . NA creates a MemoryStore . Remote stores open read-only. See vignette("remote-stores").
mode	: 'r', 'r+', 'a', 'w', 'w-', optional Persistence mode: 'r' means read only (must exist); 'r+' means read/write (must exist); 'a' means read/write (create if doesn't exist); 'w' means create (overwrite if exists); 'w-' means create (fail if exists).
path	: string, bytes, optional Path under which array is stored.
...	Additional arguments to pass to zarr_open_array or zarr_open_group.

Value

ZarrArray or ZarrGroup

zarr_open_array	<i>Open an array using file-mode-like semantics.</i>
-----------------	--

Description

Open an array using file-mode-like semantics.

Usage

```
zarr_open_array(
    store = NA,
    storage_options = NA,
    mode = NA,
    shape = NA,
    chunks = TRUE,
    dtype = NA,
    compressor = NA,
    fill_value = NA,
    order = NA,
    synchronizer = NA,
    overwrite = FALSE,
    path = NA,
    chunk_store = NA,
    filters = NA,
    cache_metadata = TRUE,
    cache_attrs = TRUE,
    object_codec = NA,
    dimension_separator = NA,
    write_empty_chunks = TRUE,
    zarr_format = 2L
)
```

Arguments

store : Store A mapping that supports string keys and bytes-like values.

storage_options : Accepted for compatibility with the zarr-python signature but **currently ignored**. pizarr configures remote stores with environment variables rather than a per-call options list; see [S3Store](#) and `vignette("remote-stores")`.

mode : 'r', 'r+', 'a', 'w', 'w-', optional Persistence mode: 'r' means read only (must exist); 'r+' means read/write (must exist); 'a' means read/write (create if doesn't exist); 'w' means create (overwrite if exists); 'w-' means create (fail if exists).

shape	: int or tuple of ints Array shape.
chunks	: bool, int or tuple of ints, optional Chunk shape. If True, will be guessed from shape and dtype. If False, will be set to shape, i.e., single chunk for the whole array.
dtype	: string or dtype, optional NumPy dtype.
compressor	: Codec, optional Primary compressor.
fill_value	: object Default value to use for uninitialized portions of the array.
order	: 'C', 'F', optional Memory layout to be used within each chunk.
synchronizer	: object, optional Array synchronizer.
overwrite	: bool, optional If True, erase all data in store prior to initialisation.
path	: string, bytes, optional Path under which array is stored.
chunk_store	: Store, optional Separate storage for chunks. If not provided, store will be used for storage of both chunks and metadata.
filters	: sequence, optional Sequence of filters to use to encode chunk data prior to compression.
cache_metadata	: bool, optional If True, array configuration metadata will be cached for the lifetime of the object. If False, array metadata will be reloaded prior to all data access and modification operations (may incur overhead depending on storage and data access pattern).
cache_attrs	: bool, optional If True (default), user attributes will be cached for attribute read operations. If False, user attributes are reloaded from the store prior to all attribute read operations.
object_codec	: Codec, optional A codec to encode object arrays, only needed if dtype=object.
dimension_separator	: '.', '/', optional Separator placed between the dimensions of a chunk.
write_empty_chunks	: bool, optional If True (default), all chunks will be stored regardless of their contents. If False, each chunk is compared to the array's fill value prior to storing. If a chunk is uniformly equal to the fill value, then that chunk is not be stored, and the store entry for that chunk's key is deleted. This setting enables sparser storage, as only chunks with non-fill-value data are stored, at the expense of overhead associated with checking the data of each chunk.
zarr_format	: int, optional Zarr format version. Use 2L (default) for Zarr V2 or 3L for Zarr V3.

Value

ZarrArray

zarr_open_group	<i>Open a group using file-mode-like semantics.</i>
-----------------	---

Description

Open a group using file-mode-like semantics.

Usage

```
zarr_open_group(
    store = NA,
    mode = NA,
    cache_attrs = TRUE,
    synchronizer = NA,
    path = NA,
    chunk_store = NA,
    storage_options = NA,
    zarr_format = 2L
)
```

Arguments

store	A Store object, or a character string that is coerced to one: an http:// or https:// URL becomes an HttpStore , an s3:// URL becomes an S3Store , a gs:// URL becomes a GcsStore , and anything else is treated as a filesystem path and becomes a DirectoryStore . NA creates a MemoryStore . Remote stores open read-only. See <code>vignette("remote-stores")</code> .
mode	: 'r', 'r+', 'a', 'w', 'w-', optional Persistence mode: 'r' means read only (must exist); 'r+' means read/write (must exist); 'a' means read/write (create if doesn't exist); 'w' means create (overwrite if exists); 'w-' means create (fail if exists).
cache_attrs	: bool, optional If True (default), user attributes will be cached for attribute read operations. If False, user attributes are reloaded from the store prior to all attribute read operations.
synchronizer	: object, optional Array synchronizer.
path	: string, bytes, optional Path under which array is stored.
chunk_store	: Store, optional Separate storage for chunks. If not provided, store will be used for storage of both chunks and metadata.
storage_options	Accepted for compatibility with the zarr-python signature but currently ignored . pizzarr configures remote stores with environment variables rather than a per-call options list; see S3Store and <code>vignette("remote-stores")</code> .
zarr_format	: int, optional Zarr format version. Use 2L (default) for Zarr V2 or 3L for Zarr V3.

Value

ZarrGroup

zarr_save_array	<i>Convenience function to save a ZarrArray to the local file system.</i>
-----------------	---

Description

Convenience function to save a ZarrArray to the local file system.

Usage

```
zarr_save_array(store, arr, ...)
```

Arguments

store	: Store A mapping that supports string keys and bytes-like values.
arr	: ZarrArray The array with data to save.
...	Additional arguments to pass to zarr_create_array().

zarr_volcano	<i>Create a demo Zarr group containing R's volcano dataset</i>
--------------	--

Description

Writes the [volcano](#) matrix into a temporary DirectoryStore as a Zarr array named "volcano" and returns the opened group.

Usage

```
zarr_volcano()
```

Value

A ZarrGroup containing a single array called "volcano".

Examples

```
g <- zarr_volcano()
v <- g$get_item("volcano")
image(v$get_item("...")$data, main = "Maunga Whau Volcano")
```

zb_int	<i>Convenience function for the internal Int class constructor with zero-based indexing</i>
--------	---

Description

Convenience function for the internal Int class constructor with zero-based indexing

Usage

```
zb_int(index)
```

Arguments

index	integer index
-------	---------------

zb_slice	<i>Convenience function for the internal Slice R6 class constructor with zero-based indexing and exclusive stop index.</i>
----------	--

Description

Convenience function for the internal Slice R6 class constructor with zero-based indexing and exclusive stop index.

Usage

```
zb_slice(start, stop = NA, step = NA)
```

Arguments

start	The start index.
stop	The stop index.
step	The step size. Negative values reverse the direction of the slice, matching Python/NumPy semantics (e.g., step = -1 iterates backward from start toward stop).

Value

A Slice instance with the specified parameters.

Examples

```
# Equivalent to Python's arr[5:0:-1]
zb_slice(5, 0, -1)

# Equivalent to Python's arr[-1::-1] (full reverse)
zb_slice(-1, NA, -1)
```

ZlibCodec

ZlibCodec Class

Description

Class representing a zlib compressor

Format

[R6::R6Class](#) inheriting from [Codec](#).

Details

Zlib compressor for Zarr

Super class

[pizzarr::Codec](#) -> ZlibCodec

Public fields

level The compression level.

Methods

Public methods:

- [ZlibCodec\\$new\(\)](#)
- [ZlibCodec\\$encode\(\)](#)
- [ZlibCodec\\$decode\(\)](#)
- [ZlibCodec\\$get_config\(\)](#)
- [ZlibCodec\\$clone\(\)](#)

Method [new\(\)](#): Create a new Zlib compressor.

Usage:

```
ZlibCodec$new(level = 6, ...)
```

Arguments:

level The compression level, between 1 and 22.

... Not used.

Returns: A new ZlibCodec object.

Method [encode\(\)](#): Compress data.

Usage:

```
ZlibCodec$encode(buf, zarr_arr)
```

Arguments:

buf (raw())
The un-compressed data.

zarr_arr ([ZarrArray](#))
The ZarrArray instance.

Returns: Compressed data.

Method decode(): Decompress data.

Usage:

ZlibCodec.decode(buf, zarr_arr)

Arguments:

buf (raw())
The compressed data.

zarr_arr ([ZarrArray](#))
The ZarrArray instance.

Returns: Un-compressed data.

Method get_config(): Get codec configuration as a list.

Usage:

ZlibCodec.get_config()

Returns: A named list.

Method clone(): The objects of this class are cloneable with this method.

Usage:

ZlibCodec.clone(deep = FALSE)

Arguments:

deep Whether to make a deep clone.

See Also

Other Codec classes: [BloscCodec](#), [Bz2Codec](#), [Codec](#), [GzipCodec](#), [Lz4Codec](#), [LzmaCodec](#), [VLenUtf8Codec](#), [ZstdCodec](#)

Index

- * **Codec classes**
 - BloscCodec, [6](#)
 - Bz2Codec, [8](#)
 - Codec, [9](#)
 - GzipCodec, [12](#)
 - LzmaCodec, [19](#)
 - VLenUtf8Codec, [26](#)
 - ZlibCodec, [38](#)
- * **Store classes**
 - HttpStore, [14](#)
- * **datasets**
 - pizzarr_option_defaults, [24](#)
- as_scalar, [2](#)
- Attributes, [3](#)
- BloscCodec, [6](#), [9](#), [10](#), [14](#), [20](#), [27](#), [39](#)
- Bz2Codec, [7](#), [8](#), [10](#), [14](#), [20](#), [27](#), [39](#)
- Codec, [6–9](#), [9](#), [12](#), [14](#), [19](#), [20](#), [26](#), [27](#), [38](#), [39](#)
- DirectoryStore, [16](#), [32](#), [35](#)
- Dtype, [11](#)
- GcsStore, [14](#), [16](#), [32](#), [35](#)
- GzipCodec, [7](#), [9](#), [10](#), [12](#), [20](#), [27](#), [39](#)
- HttpStore, [14](#), [32](#), [35](#)
- int, [17](#)
- is_key_error, [17](#)
- is_scalar, [18](#)
- is_slice, [18](#)
- Lz4Codec, [7](#), [9](#), [10](#), [14](#), [20](#), [27](#), [39](#)
- LzmaCodec, [7](#), [9](#), [10](#), [14](#), [19](#), [27](#), [39](#)
- MemoryStore, [16](#), [32](#), [35](#)
- NestedArray, [20](#)
- pizzarr::Codec, [6](#), [8](#), [13](#), [19](#), [26](#), [38](#)
- pizzarr::Store, [14](#)
- pizzarr_compiled_features, [22](#)
- pizzarr_config, [23](#)
- pizzarr_option_defaults, [24](#)
- pizzarr_sample, [24](#)
- pizzarr_upgrade, [25](#)
- R6::R6Class, [3](#), [6](#), [8](#), [9](#), [11](#), [12](#), [14](#), [19](#), [20](#), [26](#), [38](#)
- S3Store, [14](#), [16](#), [32](#), [33](#), [35](#)
- slice, [25](#)
- Store, [4](#), [14](#), [16](#), [32](#), [35](#)
- VLenUtf8Codec, [7](#), [9](#), [10](#), [14](#), [20](#), [26](#), [39](#)
- volcano, [36](#)
- zarr_create, [28](#)
- zarr_create_array, [30](#)
- zarr_create_empty, [30](#)
- zarr_create_group, [31](#)
- zarr_create_zeros, [32](#)
- zarr_open, [32](#)
- zarr_open_array, [33](#)
- zarr_open_group, [35](#)
- zarr_save_array, [36](#)
- zarr_volcano, [36](#)
- ZarrArray, [7](#), [9](#), [10](#), [13](#), [20](#), [27](#), [39](#)
- zarrs_compiled_features, [28](#)
- zb_int, [37](#)
- zb_slice, [37](#)
- ZlibCodec, [7](#), [9](#), [10](#), [14](#), [20](#), [27](#), [38](#)
- ZstdCodec, [7](#), [9](#), [10](#), [12](#), [14](#), [20](#), [27](#), [39](#)