

Package: Ropendal (via r-universe)

July 21, 2026

Type Package

Title Abstract Filesystem Access for R via 'Apache OpenDAL'

Version 0.0.1.9000

Description Provides a byte-oriented abstract filesystem interface for R backed by the Rust crate of 'Apache OpenDAL' [<https://opendal.apache.org/>](https://opendal.apache.org/). The package is designed around filesystem primitives, asynchronous Aio-like operations inspired by 'nanonext', pluggable raw-vector serializers, and a native C API for direct async byte access by other R packages.

License GPL (>= 3)

URL <https://github.com/sounkou-bioinfo/Ropendal>,
<https://sounkou-bioinfo.github.io/Ropendal>

BugReports <https://github.com/sounkou-bioinfo/Ropendal/issues>

SystemRequirements Cargo (Rust's package manager), rustc >= 1.88.0,
GNU make

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Imports S7

Suggests tinytest, roxygen2, rmarkdown, knitr, pkgdown, Rtinycc

VignetteBuilder knitr

Config/pak/sysreqs make libclang-dev

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

Date/Publication 2026-07-06 14:28:27 UTC

RemoteUrl <https://github.com/RGenomicsETL/Ropendal>

RemoteRef HEAD

RemoteSha 3fb2f41e49966a1cf594762cf1a9414b248ba430

Contents

byte-store	2
OpendalAio	6
OpendalCredentialProvider	6
OpendalFs	7
OpendalHttpFixture	7
OpendalLsIter	7
OpendalReadIter	8
OpendalWriteIter	8
Ropendal-api	9
Index	16

byte-store	<i>Byte store adapter</i>
------------	---------------------------

Description

A small key-to-bytes adapter over an OpendalFs prefix. It is intended as a substrate for Zarr-like chunk layouts and other index-driven readers that want store-relative keys without leaving Ropendal’s byte-first filesystem API.

Usage

```
byte_store(fs, prefix = "")

store_cache(
    store,
    cache_dir = tools::R_user_dir("Ropendal", "cache"),
    validate = c("last_modified_size", "none")
)

store_block_cache(
    store,
    cache_dir = tools::R_user_dir("Ropendal", "cache"),
    block_size = 8 * 1024^2,
    validate = c("last_modified_size", "none")
)

store_cache_clear(store)

store_read(
    store,
    key,
    mode = c("bytes", "raw"),
    offset = NULL,
    size = NULL,
```

```
    end = NULL,  
    result = c("auto", "flat", "nested"),  
    batch_concurrency = NULL,  
    read_concurrency = NULL,  
    chunk_size = NULL,  
    coalesce_gap = NULL  
)  
  
store_read_aio(  
    store,  
    key,  
    mode = c("bytes", "raw"),  
    offset = NULL,  
    size = NULL,  
    end = NULL,  
    result = c("auto", "flat", "nested"),  
    batch_concurrency = NULL,  
    read_concurrency = NULL,  
    chunk_size = NULL,  
    coalesce_gap = NULL  
)  
  
store_write(  
    store,  
    key,  
    data,  
    batch_concurrency = NULL,  
    write_concurrency = NULL,  
    chunk_size = NULL  
)  
  
store_write_aio(  
    store,  
    key,  
    data,  
    batch_concurrency = NULL,  
    write_concurrency = NULL,  
    chunk_size = NULL  
)  
  
store_replace(  
    store,  
    key,  
    data,  
    batch_concurrency = NULL,  
    write_concurrency = NULL,  
    chunk_size = NULL  
)
```

```

store_replace_aio(
    store,
    key,
    data,
    batch_concurrency = NULL,
    write_concurrency = NULL,
    chunk_size = NULL
)

store_exists(store, key, batch_concurrency = NULL)

store_exists_aio(store, key, batch_concurrency = NULL)

store_list(
    store,
    path = "",
    recursive = FALSE,
    limit = NULL,
    start_after = NULL
)

store_list_aio(
    store,
    path = "",
    recursive = FALSE,
    limit = NULL,
    start_after = NULL
)

store_delete(store, key, recursive = FALSE, batch_concurrency = NULL)

store_delete_aio(store, key, recursive = FALSE, batch_concurrency = NULL)

```

Arguments

<code>fs</code>	Ropendal filesystem handle.
<code>prefix</code>	Root-relative filesystem prefix used as the store root.
<code>store</code>	Byte store object returned by <code>byte_store()</code> , <code>store_cache()</code> , or <code>store_block_cache()</code> .
<code>cache_dir</code>	Local directory used by <code>store_cache()</code> or <code>store_block_cache()</code> .
<code>validate</code>	Cache validation strategy. "last_modified_size" compares parent object size and modification time before using a cached value. "none" skips modification-time checks; block caches still stat the parent to bound ranges and refresh when cached block metadata has a different object size.
<code>block_size</code>	Fixed block size in bytes for <code>store_block_cache()</code> .
<code>key</code>	Store-relative object key or keys.
<code>mode</code>	Read materialization: <code>OpendalBytes</code> handles or raw vectors.

offset, size, end	Optional byte-range controls forwarded to the read operation.
result	Requested read result shape.
batch_concurrency	Optional maximum number of keys to process concurrently.
read_concurrency, write_concurrency	Optional per-object OpenDAL transfer concurrency.
chunk_size	Optional transfer chunk size in bytes.
coalesce_gap	Optional byte gap for coalescing nearby read ranges.
data	Raw vector, <code>OpendalBytes</code> , or list of raw vectors/byte handles.
path	Store-relative directory path for <code>store_list()</code> .
recursive	Whether to list or delete recursively.
limit	Optional maximum listing entries.
start_after	Optional store-relative listing continuation marker.

Details

`store_read()` returns `OpendalBytes` by default so callers can stay below ordinary R raw-vector materialization. `mode = "raw"` or `as.raw()` exposes the raw-vector API; for `OpendalBytes`, `as.raw()` returns a raw ALTREP facade that materializes only when R requires raw-vector pointer access. The store layer is intentionally byte-only: serializers, deserializers, codecs, and format parsers belong above it.

`store_cache()` wraps a byte store in an explicit local full-object cache. It is useful for chunked layouts where each key is already a small block. Partial byte-range reads and non-default read shaping bypass this first cache layer. `store_block_cache()` wraps a byte store in an explicit fixed-size block cache for scalar complete or range reads with `result = "auto"` while keeping format semantics above the store layer. Vectorized ranges, vectorized keys, and non-auto result shapes bypass this cache adapter and read from the parent store directly.

Value

`byte_store()`, `store_cache()`, and `store_block_cache()` return byte-store objects. Store operations return the corresponding filesystem values with paths rewritten relative to the store root for listings.

Examples

```
root <- tempfile("ropendal-store-")
dir.create(root)
fs <- opendal("fs", root = root)
store <- byte_store(fs, "array.zarr")
store_write(store, "zarr.json", charToRaw("{}"))
as.raw(store_read(store, "zarr.json"))
cached <- store_cache(store, tempfile("ropendal-cache-"), validate = "none")
as.raw(store_read(cached, "zarr.json"))
```

OpendalAio	<i>Asynchronous operation handle.</i>
------------	---------------------------------------

Description

Asynchronous operation handle.

Usage

OpendalAio

Format

An object of class Ropendal::OpendalAio__bundle (inherits from savvy_Ropendal__sealed) of length 0.

OpendalCredentialProvider	<i>Explicit credential provider.</i>
---------------------------	--------------------------------------

Description

Explicit credential provider.

Usage

OpendalCredentialProvider

Format

An object of class Ropendal::OpendalCredentialProvider__bundle (inherits from savvy_Ropendal__sealed) of length 5.

OpendalFs	<i>Filesystem handle backed by Apache OpenDAL.</i>
-----------	--

Description

Filesystem handle backed by Apache OpenDAL.

Usage

OpendalFs

Format

An object of class `Ropendal::OpendalFs__bundle` (inherits from `savvy_Ropendal__sealed`) of length 2.

OpendalHttpFixture	<i>Internal HTTP fixture for tests.</i>
--------------------	---

Description

Internal HTTP fixture for tests.

Usage

OpendalHttpFixture

Format

An object of class `Ropendal::OpendalHttpFixture__bundle` (inherits from `savvy_Ropendal__sealed`) of length 1.

OpendalLsIter	<i>Streaming listing iterator over one prefix.</i>
---------------	--

Description

Streaming listing iterator over one prefix.

Usage

OpendalLsIter

Format

An object of class `Ropendal::OpendalLsIter__bundle` (inherits from `savvy_Ropendal__sealed`) of length 0.

OpendalReadIter	<i>Chunked read iterator over one object.</i>
-----------------	---

Description

Chunked read iterator over one object.

Usage

OpendalReadIter

Format

An object of class Ropendal::OpendalReadIter__bundle (inherits from savvy_Ropendal__sealed) of length 0.

OpendalWriteIter	<i>Chunked write sink for one object.</i>
------------------	---

Description

Chunked write sink for one object.

Usage

OpendalWriteIter

Format

An object of class Ropendal::OpendalWriteIter__bundle (inherits from savvy_Ropendal__sealed) of length 0.

Description

Byte-oriented filesystem operations backed by Apache OpenDAL. The current implementation includes local fs, HTTP, S3-compatible, and Google Drive handles; raw byte operations; metadata and listing where supported; error values; Aio handles for read, write, metadata, and namespace operations; and a pure C API.

Usage

```

ropendal(scheme = "fs", ..., root = NULL, config = list(), auth = NULL,
         headers = NULL, runtime = runtime_config(), layers = list())
ropendal_uri(uri, headers = NULL, runtime = runtime_config(), layers = list())
credentials_s3(access_key_id, secret_access_key,
               session_token = "", region = "", source = "direct")
credentials_gcs(token = "", service_account_key = "",
                credential_path = "", scope = "", source = "direct")
credentials_azblob(account_name = "", account_key = "", sas_token = "",
                   endpoint = "", source = "direct")
credentials_gdrive(access_token = "", refresh_token = "",
                   client_id = "", client_secret = "", source = "direct")
credentials_gdrive3(secret_json,
                    tokens_json = file.path(dirname(secret_json), "tokens.json"),
                    scope = "https://www.googleapis.com/auth/drive")
credential_schemes(provider)
credential_config(provider, service)
credential_summary(provider)
runtime_config(threads = NULL)
layer_concurrent_limit(max)
layer_timeout(request_timeout = NULL, io_timeout = NULL)
fs_info(fs)
fs_capabilities(fs)
fs_normalize_path(fs, path, directory = FALSE)
byte_ranges(path, offset, size = NULL, end = NULL, id = NULL, result = "flat")
opt(fs, name)
opt(fs, name) <- value
serial_config(class, sfunc, ufunc)
codec_config(name, class = "raw", sfunc = NULL, ufunc = NULL)
serialize_raw(x, config = list())
deserialize_raw(x, config = list())
fs_read(fs, path, offset = 0, size = NULL, end = NULL,
        result = c("auto", "flat", "nested"), batch_concurrency = NULL,
        read_concurrency = NULL, chunk_size = NULL, coalesce_gap = NULL,
        mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
        serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))

```

```

fs_read_aio(fs, path, offset = 0, size = NULL, end = NULL,
            result = c("auto", "flat", "nested"), batch_concurrency = NULL,
            read_concurrency = NULL, chunk_size = NULL, coalesce_gap = NULL,
            mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
            serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_read_bytes(fs, path, offset = 0, size = NULL, end = NULL,
              result = c("auto", "flat", "nested"), batch_concurrency = NULL,
              read_concurrency = NULL, chunk_size = NULL, coalesce_gap = NULL)
fs_read_bytes_aio(fs, path, offset = 0, size = NULL, end = NULL,
                  result = c("auto", "flat", "nested"), batch_concurrency = NULL,
                  read_concurrency = NULL, chunk_size = NULL, coalesce_gap = NULL)
opendal_bytes_slice(x, offset = 0, size = NULL, end = NULL)
## S3 method for class 'OpendalBytes'
as.raw(x)
## S3 method for class 'OpendalBytes'
length(x)
fs_read_iter(fs, path, chunk_size, offset = 0, size = NULL,
             read_concurrency = NULL, coalesce_gap = NULL)
read_iter_next(iter)
read_iter_collect(iter)
fs_tell(iter)
fs_seek(iter, offset, whence = c("start", "current", "end"))
fs_write(fs, path, data, batch_concurrency = NULL,
         write_concurrency = NULL, chunk_size = NULL,
         mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
         serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_write_aio(fs, path, data, batch_concurrency = NULL,
             write_concurrency = NULL, chunk_size = NULL,
             mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
             serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_replace(fs, path, data, batch_concurrency = NULL,
          write_concurrency = NULL, chunk_size = NULL,
          mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
          serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_replace_aio(fs, path, data, batch_concurrency = NULL,
              write_concurrency = NULL, chunk_size = NULL,
              mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
              serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_append(fs, path, data, batch_concurrency = NULL,
          write_concurrency = NULL, chunk_size = NULL,
          mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
          serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_append_aio(fs, path, data, batch_concurrency = NULL,
              write_concurrency = NULL, chunk_size = NULL,
              mode = c("raw", "serial", "text", "codec"), encoding = "UTF-8",
              serial_config = opt(fs, "serial"), codec = opt(fs, "codec"))
fs_write_iter(fs, path, create = TRUE, append = FALSE,
              write_concurrency = NULL, chunk_size = NULL)

```

```
write_iter_write(iter, data)
write_iter_close(iter)
fs_stat(fs, path, batch_concurrency = NULL)
fs_stat_aio(fs, path, batch_concurrency = NULL)
fs_stats(fs, path, batch_concurrency = NULL)
fs_stats_aio(fs, path, batch_concurrency = NULL)
fs_exists(fs, path, batch_concurrency = NULL)
fs_exists_aio(fs, path, batch_concurrency = NULL)
fs_ls(fs, path = "", recursive = FALSE, limit = NULL, start_after = NULL)
fs_ls_aio(fs, path = "", recursive = FALSE,
          limit = NULL, start_after = NULL)
fs_ls_iter(fs, path = "", recursive = FALSE, page_size = 1000,
           limit = NULL, start_after = NULL, prefetch = 0)
fs_walk_iter(fs, path = "", page_size = 1000,
             limit = NULL, start_after = NULL, prefetch = 0)
ls_iter_next(iter)
ls_iter_collect(iter)
walk_iter_next(iter)
walk_iter_collect(iter)
fs_mkdir(fs, path)
fs_mkdir_aio(fs, path, batch_concurrency = NULL)
fs_delete(fs, path, recursive = FALSE, batch_concurrency = NULL)
fs_delete_aio(fs, path, recursive = FALSE, batch_concurrency = NULL)
fs_copy(fs, from, to)
fs_copy_aio(fs, from, to, batch_concurrency = NULL)
fs_rename(fs, from, to)
fs_rename_aio(fs, from, to, batch_concurrency = NULL)
collect_aio(aio)
collect_aio_(aio)
call_aio(aio)
call_aio_(aio)
stop_aio(aio)
poll_aio(aio)
cv()
cv_value(cv)
cv_reset(cv)
cv_signal(cv)
cv_wait(cv, interval = 0.001)
cv_until(cv, msec = 0, interval = 0.001)
aio_monitor(aio, cv = cv())
read_monitor(monitor)
race_aio(aio, timeout = NULL, interval = 0.001)
unresolved(x = NULL)
is_error_value(x)
error_kind(x)
error_message(x)
error_operation(x)
error_path(x)
```

Arguments

scheme	OpenDAL service scheme.
...	Named service configuration entries.
root	Root path/prefix for the service.
config	For <code>opendal()</code> , named list of service configuration entries. For <code>serialize_raw()</code> / <code>deserialize_raw()</code> , serialization config.
auth	Explicit credential provider.
headers	Named scalar character list/vector of HTTP headers for http/https filesystems; values may contain credentials and are not printed by Ropendal.
runtime	Runtime configuration from <code>runtime_config()</code> .
layers	List of explicit filesystem layers, currently including <code>layer_concurrent_limit()</code> and <code>layer_timeout()</code> .
threads	Number of Tokio worker threads for a filesystem handle.
max request_timeout	Maximum total in-flight backend operations for a filesystem handle.
	Operation timeout in seconds for non-streaming backend operations such as <code>stat</code> , <code>exists</code> , <code>delete</code> , <code>copy</code> , <code>rename</code> , and <code>mkdir</code> . If only one timeout field is supplied, the omitted field uses OpenDAL's timeout-layer default.
io_timeout	I/O timeout in seconds for backend read/write streams and listing iteration. If only one timeout field is supplied, the omitted field uses OpenDAL's timeout-layer default.
uri	OpenDAL URI.
access_key_id, secret_access_key	S3-compatible access key fields.
session_token	Optional S3-compatible session token.
region	Optional S3-compatible signing region.
token	Google Cloud Storage OAuth2 bearer token.
service_account_key	Google Cloud Storage service-account JSON string.
credential_path	Explicit path to Google Cloud Storage credential JSON.
account_name	Azure Blob Storage account name.
account_key	Azure Blob Storage account key.
sas_token	Azure Blob Storage SAS token.
endpoint	Optional Azure Blob Storage endpoint.
access_token	Google Drive access token, mutually exclusive with <code>refresh_token</code> .
refresh_token	Google Drive refresh token.
client_id, client_secret	OAuth client fields required with <code>refresh_token</code> .
source	Redacted source label for a credential provider.

<code>secret_json</code>	Path to a JSON file containing Google Drive <code>client_id</code> and <code>client_secret</code> .
<code>tokens_json</code>	Path to a token JSON file containing a refresh token.
<code>scope</code>	Scope used to choose a refresh token from <code>tokens_json</code> .
<code>provider</code>	Credential provider object.
<code>service</code>	OpenDAL service scheme for credential materialization.
<code>fs</code>	Ropendal filesystem handle.
<code>path</code>	Root-relative path or paths. For <code>fs_read()</code> , <code>fs_read_aio()</code> , <code>fs_read_bytes()</code> , and <code>fs_read_bytes_aio()</code> , may also be a <code>byte_ranges()</code> request object.
<code>directory</code>	Whether to normalize as a directory path.
<code>offset</code>	Zero-based byte offset or offsets. Omit or use <code>NULL</code> to read each path from byte zero; explicit numeric offsets are not recycled across multiple paths. Use numeric vectors for many ranges from one path and lists of numeric vectors for per-path ranges.
<code>size</code>	Number of bytes to read, or <code>NULL</code> to read to EOF. When supplied, <code>size</code> follows the same scalar/vector/list shape as <code>offset</code> .
<code>end</code>	Exclusive byte end offset. When supplied, <code>end</code> follows the same scalar/vector/list shape as <code>offset</code> .
<code>id</code>	Optional identifiers for ranges in a <code>byte_ranges()</code> request. Flat request results are named with <code>id</code> when lengths match; nested result names require a list-shaped <code>id</code> aligned with per-path ranges.
<code>result</code>	Requested result shape.
<code>name</code>	Option name.
<code>value</code>	Option value.
<code>class</code>	Class name or names matched for custom serialization.
<code>sfunc, ufunc</code>	Serializer and deserializer functions for <code>serial_config()</code> . R-closure codecs are not supported; codec transforms are native byte transforms.
<code>mode</code>	Materialization mode: raw bytes, text strings, serialized R objects, or raw bytes passed through an explicit codec.
<code>encoding</code>	Text encoding for <code>mode = "text"</code> reads and writes. Encodings whose encoded bytes contain NUL bytes are rejected; use raw mode for those storage formats.
<code>serial_config</code>	Serialization config, normally <code>opt(fs, "serial")</code> .
<code>codec</code>	Optional native byte codec name or <code>codec_config()</code> object, normally <code>opt(fs, "codec")</code> .
<code>batch_concurrency</code>	Optional maximum number of independent paths/ranges to process concurrently. <code>NULL</code> uses the Ropendal default; explicit 0 is invalid in the R API.
<code>read_concurrency</code>	Optional per-object OpenDAL read concurrency for large reads where the backend supports chunked/concurrent reads.
<code>write_concurrency</code>	Optional per-object OpenDAL write concurrency for large writes where the backend supports multipart/concurrent writes.

<code>chunk_size</code>	Optional read/write chunk size in bytes for OpenDAL's per-object transfer planning.
<code>coalesce_gap</code>	Optional byte gap for coalescing nearby read ranges.
<code>data</code>	Raw vector, <code>OpendalBytes</code> handle, or list mixing raw vectors and <code>OpendalBytes</code> handles for multiple paths in raw mode; arbitrary R object or list of objects in serial mode.
<code>iter</code>	Read, write, listing, or walking iterator handle.
<code>whence</code>	Seek origin for read iterators: <code>iterator start</code> , <code>current</code> , or <code>end</code> .
<code>create</code>	Whether a write iterator should create only and fail if the target exists.
<code>append</code>	Whether a write iterator should append rather than replace.
<code>recursive</code>	Whether to recurse for operations that support it.
<code>limit</code>	Optional maximum number of listing entries to materialize or return across an iterator.
<code>start_after</code>	Optional root-relative listing continuation marker; entries less than or equal to this path are skipped where supported. Iterator <code>\$cursor</code> values are last yielded paths and can be used as best-effort <code>start_after</code> markers for lexically ordered listings; they are not opaque backend continuation tokens.
<code>page_size</code>	Maximum number of entries returned by one iterator page.
<code>prefetch</code>	Number of listing entries an iterator may buffer ahead in Rust/Tokio. Positive values may start bounded background prefetch at iterator construction; use <code>0</code> to keep strict demand-driven iteration.
<code>from, to</code>	Source and destination paths.
<code>aio</code>	Aio handle.
<code>cv</code>	Condition variable object from <code>cv()</code> .
<code>monitor</code>	Monitor handle from <code>aio_monitor()</code> .
<code>msec</code>	Milliseconds to wait for <code>cv_until()</code> .
<code>timeout</code>	Optional milliseconds to wait for <code>race_aio()</code> .
<code>interval</code>	Poll interval in seconds for R-thread wait helpers.
<code>x</code>	Object to inspect.

Details

Filesystem failures are returned as classed error values. Invalid arguments and internal runtime failures may signal R errors. Credential helpers return classed providers with redacted printing; pass them with `auth =`.

Aio handles expose read-only active bindings. `$value` returns an `unresolvedValue` while the operation is pending and the resolved value or error value afterwards. `$data` and `$result` are aliases for `$value`, while `$state`, `$resolved`, and `$error` expose readiness and error inspection. `collect_aio()` waits and returns the value. `call_aio()` waits, updates the Aio, and returns the Aio invisibly. `unresolved()` constructs the unresolved sentinel; `unresolved(aio)` and `unresolved(value)` are predicates.

`byte_ranges()` builds a small range request object for index-heavy formats. Pass it as the path argument to `fs_read()`, `fs_read_aio()`, `fs_read_bytes()`, or `fs_read_bytes_aio()`; the object

is unpacked into the same vector/list path, offset, size, and end shapes as direct calls. Request objects default to `result = "flat"`, which is convenient for row-oriented index tables.

`opendal_bytes_slice()` returns another immutable `OpendalBytes` handle for a byte subrange without first materializing the complete handle as an R raw vector. `offset` is zero-based and `end` is exclusive, matching read ranges. `as.raw()` on an `OpendalBytes` handle returns a raw ALTREP facade: element and region access copy only requested bytes, while pointer access materializes an R-owned raw vector.

Value

Filesystem handles, raw vectors, `OpendalBytes` handles, deserialized R objects, metadata lists, logical results, Aio handles, or classed error values depending on the operation and mode.

Index

* datasets

- [OpendalAio](#), 6
- [OpendalCredentialProvider](#), 6
- [OpendalFs](#), 7
- [OpendalHttpFixture](#), 7
- [OpendalLsIter](#), 7
- [OpendalReadIter](#), 8
- [OpendalWriteIter](#), 8

[aio_monitor](#) (Ropendal-api), 9

[as.raw.OpendalBytes](#) (Ropendal-api), 9

[byte-store](#), 2

[byte_ranges](#) (Ropendal-api), 9

[byte_store](#) (byte-store), 2

[call_aio](#) (Ropendal-api), 9

[call_aio_](#) (Ropendal-api), 9

[codec_config](#) (Ropendal-api), 9

[collect_aio](#) (Ropendal-api), 9

[collect_aio_](#) (Ropendal-api), 9

[credential_config](#) (Ropendal-api), 9

[credential_schemes](#) (Ropendal-api), 9

[credential_summary](#) (Ropendal-api), 9

[CredentialProvider](#) (Ropendal-api), 9

[credentials_azblob](#) (Ropendal-api), 9

[credentials_gcs](#) (Ropendal-api), 9

[credentials_gdrive](#) (Ropendal-api), 9

[credentials_gdrive3](#) (Ropendal-api), 9

[credentials_s3](#) (Ropendal-api), 9

[cv](#) (Ropendal-api), 9

[cv_reset](#) (Ropendal-api), 9

[cv_signal](#) (Ropendal-api), 9

[cv_until](#) (Ropendal-api), 9

[cv_value](#) (Ropendal-api), 9

[cv_wait](#) (Ropendal-api), 9

[deserialize_raw](#) (Ropendal-api), 9

[error_kind](#) (Ropendal-api), 9

[error_message](#) (Ropendal-api), 9

[error_operation](#) (Ropendal-api), 9

[error_path](#) (Ropendal-api), 9

[fs_append](#) (Ropendal-api), 9

[fs_append_aio](#) (Ropendal-api), 9

[fs_capabilities](#) (Ropendal-api), 9

[fs_copy](#) (Ropendal-api), 9

[fs_copy_aio](#) (Ropendal-api), 9

[fs_delete](#) (Ropendal-api), 9

[fs_delete_aio](#) (Ropendal-api), 9

[fs_exists](#) (Ropendal-api), 9

[fs_exists_aio](#) (Ropendal-api), 9

[fs_info](#) (Ropendal-api), 9

[fs_ls](#) (Ropendal-api), 9

[fs_ls_aio](#) (Ropendal-api), 9

[fs_ls_iter](#) (Ropendal-api), 9

[fs_mkdir](#) (Ropendal-api), 9

[fs_mkdir_aio](#) (Ropendal-api), 9

[fs_normalize_path](#) (Ropendal-api), 9

[fs_read](#) (Ropendal-api), 9

[fs_read_aio](#) (Ropendal-api), 9

[fs_read_bytes](#) (Ropendal-api), 9

[fs_read_bytes_aio](#) (Ropendal-api), 9

[fs_read_iter](#) (Ropendal-api), 9

[fs_rename](#) (Ropendal-api), 9

[fs_rename_aio](#) (Ropendal-api), 9

[fs_replace](#) (Ropendal-api), 9

[fs_replace_aio](#) (Ropendal-api), 9

[fs_seek](#) (Ropendal-api), 9

[fs_stat](#) (Ropendal-api), 9

[fs_stat_aio](#) (Ropendal-api), 9

[fs_stats](#) (Ropendal-api), 9

[fs_stats_aio](#) (Ropendal-api), 9

[fs_tell](#) (Ropendal-api), 9

[fs_walk_iter](#) (Ropendal-api), 9

[fs_write](#) (Ropendal-api), 9

[fs_write_aio](#) (Ropendal-api), 9

[fs_write_iter](#) (Ropendal-api), 9

[is_error_value](#) (Ropendal-api), 9

layer_concurrent_limit (Ropendal-api), 9
layer_timeout (Ropendal-api), 9
length.OpendalBytes (Ropendal-api), 9
ls_iter_collect (Ropendal-api), 9
ls_iter_next (Ropendal-api), 9

opendal (Ropendal-api), 9
opendal_bytes_slice (Ropendal-api), 9
opendal_uri (Ropendal-api), 9
OpendalAio, 6
OpendalBytes (Ropendal-api), 9
OpendalCredentialProvider, 6
OpendalFs, 7
OpendalHttpFixture, 7
OpendalLsIter, 7
OpendalReadIter, 8
OpendalWriteIter, 8
opt (Ropendal-api), 9
opt<- (Ropendal-api), 9

poll_aio (Ropendal-api), 9

race_aio (Ropendal-api), 9
read_iter_collect (Ropendal-api), 9
read_iter_next (Ropendal-api), 9
read_monitor (Ropendal-api), 9
Ropendal-api, 9
runtime_config (Ropendal-api), 9

serial_config (Ropendal-api), 9
serialize_raw (Ropendal-api), 9
stop_aio (Ropendal-api), 9
store_block_cache (byte-store), 2
store_cache (byte-store), 2
store_cache_clear (byte-store), 2
store_delete (byte-store), 2
store_delete_aio (byte-store), 2
store_exists (byte-store), 2
store_exists_aio (byte-store), 2
store_list (byte-store), 2
store_list_aio (byte-store), 2
store_read (byte-store), 2
store_read_aio (byte-store), 2
store_replace (byte-store), 2
store_replace_aio (byte-store), 2
store_write (byte-store), 2
store_write_aio (byte-store), 2

unresolved (Ropendal-api), 9

walk_iter_collect (Ropendal-api), 9
walk_iter_next (Ropendal-api), 9
write_iter_close (Ropendal-api), 9
write_iter_write (Ropendal-api), 9