Package: zstd (via r-universe)

October 21, 2026

Title Zstandard Compression

Version 0.1.0

Description Compress and decompress data with the 'Zstandard' algorithm <https://facebook.github.io/zstd/>, a fast lossless compression format. Works on raw vectors in memory and on files, and can create and extract 'zstd' compressed 'tar' archives. Supports compression levels, multithreaded compression, long distance matching, content checksums, and training dictionaries for better compression of small, similar inputs.

License MIT + file LICENSE

Copyright This package includes code from the 'Zstandard' library owned by Meta Platforms, Inc. and affiliates. and created by Yann Collet, and code from the 'microtar' library created by rxi. See file 'inst/COPYRIGHTS' for details.

URL https://github.com/r-lib/zstd

BugReports https://github.com/r-lib/zstd/issues

Encoding UTF-8

Roxygen list(markdown = TRUE)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Config/roxygen2/version 8.1.0

Repository https://r-lib.r-universe.dev

Date/Publication 2026-10-08 18:39:17 UTC

RemoteUrl https://github.com/r-lib/zstd

RemoteRef HEAD

RemoteSha c935e335ffbb1ad77b1bdcda02f41476ab54533b

Contents

zstd_compress 2
zstd_decompress 3
zstd_info 4
zstd_mem_compress 5
zstd_mem_decompress 7
zstd_min_clevel 7
zstd_tar_compress 8
zstd_tar_decompress 10
zstd_train_dict 11

Index **12**

`zstd_compress` *Compress a file with Zstandard*

Description

Uses the streaming API, so memory use stays bounded regardless of the size of `input`.

Usage

```
zstd_compress(
  input,
  output,
  level = zstd_default_clevel(),
  dict = NULL,
  window_log = NULL,
  checksum = FALSE,
  strategy = NULL,
  nb_workers = 0L,
  content_size = TRUE,
  dict_id = TRUE,
  long_distance_matching = NULL
)
```

Arguments

`input`	Path to the file to compress.
`output`	Path of the compressed file to create. Overwritten if it already exists.
`level`	Integer compression level. Defaults to `zstd_default_clevel()`, which is 3 on this system. Valid range is `zstd_min_clevel()` to `zstd_max_clevel()`, which is -131072 to 22 on this system.
`dict`	`NULL`, or a raw vector containing a dictionary (as created by `zstd_train_dict()`, or any raw content dictionary). The same dictionary must be passed to `zstd_decompress()`.

window_log
: NULL (library default), or an integer setting the maximum back-reference distance as a power of two, in bytes. Larger values can improve the compression ratio of large, redundant inputs, at the cost of memory use on both the compression and decompression side.

checksum
: Whether to store a checksum of the decompressed content in the frame. zstd_mem_decompress() and zstd_decompress() always verify this checksum automatically when present, and error if it does not match.

strategy
: NULL (library default for level), or one of "fast", "dfast", "greedy", "lazy", "lazy2", "btlazy2", "btopt", "btultra", "btultra2", from fastest to strongest.

nb_workers
: Number of compression worker threads. 0 (the default) compresses on the calling thread. Higher values can speed up compression of large inputs at some cost to the compression ratio.

content_size
: Whether to store the decompressed size in the frame header, when known. Defaults to TRUE.

dict_id
: Whether to store the dictionary's ID in the frame header, when a dictionary is used. Defaults to TRUE.

long_distance_matching
: NULL (library default), or TRUE/ FALSE to force long distance matching on or off. Improves the compression ratio of large inputs with repetition far apart.

Value

output, invisibly.

Examples

```
src <- tempfile()
dst <- tempfile()
writeLines(paste(rep("hello world", 1000), collapse = " "), src)
zstd_compress(src, dst)
zstd_decompress(dst, src2 <- tempfile())
identical(
  readBin(src, "raw", file.size(src)),
  readBin(src2, "raw", file.size(src2))
)
unlink(c(src, dst, src2))
```

---

zstd_decompress *Decompress a Zstandard-compressed file*

Description

Uses the streaming API, so memory use stays bounded regardless of the size of input. If the frame carries a content checksum (see the checksum argument of zstd_compress()), it is verified automatically; an error is raised if the checksum does not match.

Usage

```
zstd_decompress(input, output, dict = NULL)
```

Arguments

input	Path to a Zstandard-compressed file.
output	Path of the decompressed file to create. Overwritten if it already exists.
dict	NULL, or a raw vector containing the dictionary that was used to compress input. See zstd_compress().

Value

output, invisibly.

Examples

```
src <- tempfile()
dst <- tempfile()
writeLines("hello world", src)
zstd_compress(src, dst)
zstd_decompress(dst, src2 <- tempfile())
identical(
  readBin(src, "raw", file.size(src)),
  readBin(src2, "raw", file.size(src2))
)
unlink(c(src, dst, src2))
```

---

zstd_info — *Get information about a Zstandard-compressed file*

Description

Reads the frame headers of a zstd file without decompressing its content. Handles concatenated files (multiple zstd frames stored back to back, e.g. produced by appending compressed chunks to an existing file), returning one row per frame in the order they appear in the file.

Usage

```
zstd_info(path)
```

Arguments

path	Path to a zstd-compressed file, or a glob pattern (e.g. "*.zst") matching several files.

Value

A data frame with one row per frame, and columns:

- `path`: path of the file the frame came from.
- `type`: `"frame"` for a regular zstd frame, `"skippable"` for a skippable frame.
- `compressed_size`: size of the frame on disk, in bytes.
- `content_size`: decompressed size of the frame, in bytes, or `NA` if unknown (streaming frames) or the frame is skippable.
- `window_size`: the decompression window size needed, in bytes, or `NA` for skippable frames.
- `dict_id`: the dictionary ID used to compress the frame, or `NA` if none was used or the frame is skippable.
- `checksum`: whether the frame includes a content checksum, or `NA` for skippable frames.

Examples

```
tmp <- tempfile()
writeBin(zstd_mem_compress(charToRaw("hello world")), tmp)
zstd_info(tmp)
unlink(tmp)
```

---

zstd_mem_compress *Compress a raw vector with Zstandard*

Description

Compress a raw vector with Zstandard

Usage

```
zstd_mem_compress(
  x,
  level = zstd_default_clevel(),
  dict = NULL,
  window_log = NULL,
  checksum = FALSE,
  strategy = NULL,
  nb_workers = 0L,
  content_size = TRUE,
  dict_id = TRUE,
  long_distance_matching = NULL
)
```

Arguments

`x`	A raw vector to compress.
`level`	Integer compression level. Defaults to `zstd_default_clevel()`, which is 3 on this system. Valid range is `zstd_min_clevel()` to `zstd_max_clevel()`, which is -131072 to 22 on this system.
`dict`	`NULL`, or a raw vector containing a dictionary (as created by `zstd_train_dict()`, or any raw content dictionary). Using a dictionary can substantially improve the compression ratio of small, similar inputs. The same dictionary must be passed to `zstd_mem_decompress()`.
`window_log`	`NULL` (library default), or an integer setting the maximum back-reference distance as a power of two, in bytes. Larger values can improve the compression ratio of large, redundant inputs, at the cost of memory use on both the compression and decompression side.
`checksum`	Whether to store a checksum of the decompressed content in the frame. `zstd_mem_decompress()` and `zstd_decompress()` always verify this checksum automatically when present, and error if it does not match.
`strategy`	`NULL` (library default for `level`), or one of `"fast"`, `"dfast"`, `"greedy"`, `"lazy"`, `"lazy2"`, `"btlazy2"`, `"btopt"`, `"btultra"`, `"btultra2"`, from fastest to strongest.
`nb_workers`	Number of compression worker threads. `0` (the default) compresses on the calling thread. Higher values can speed up compression of large inputs at some cost to the compression ratio.
`content_size`	Whether to store the decompressed size in the frame header, when known. Defaults to `TRUE`.
`dict_id`	Whether to store the dictionary's ID in the frame header, when a dictionary is used. Defaults to `TRUE`.
`long_distance_matching`	`NULL` (library default), or `TRUE`/ `FALSE` to force long distance matching on or off. Improves the compression ratio of large inputs with repetition far apart.

Value

A raw vector: the compressed data.

Examples

```
x <- charToRaw(paste(rep("hello world ", 1000), collapse = ""))
cmp <- zstd_mem_compress(x)
identical(zstd_mem_decompress(cmp), x)
```

zstd_mem_decompress *Decompress a Zstandard-compressed raw vector*

Description

If the frame carries a content checksum (see the `checksum` argument of `zstd_mem_compress()`), it is verified automatically; an error is raised if the checksum does not match.

Usage

```
zstd_mem_decompress(x, dict = NULL)
```

Arguments

`x`	A raw vector of Zstandard-compressed data (a single frame).
`dict`	`NULL`, or a raw vector containing the dictionary that was used to compress `x`. See `zstd_mem_compress()`.

Value

A raw vector: the decompressed data.

Examples

```
x <- charToRaw("hello world")
zstd_mem_decompress(zstd_mem_compress(x))
```

zstd_min_clevel *Zstandard compression level bounds*

Description

`zstd_min_clevel()` and `zstd_max_clevel()` return the smallest and largest compression levels that the bundled zstd library accepts. `zstd_default_clevel()` returns the level used when you do not specify one.

Usage

```
zstd_min_clevel()

zstd_max_clevel()

zstd_default_clevel()
```

Details

Positive levels trade speed for a better compression ratio. Levels above 19 need a lot more memory, both for compression and decompression.

Negative levels are "fast" modes. They are faster than level 1, but compress less. The smallest allowed level is very negative, because zstd uses the level to set its internal target length, and the negative of the largest target length ($2^{17} = 131072$) is the lower bound. Levels below about -7 give little extra speed, and the output quickly gets close to the size of the input, so they are rarely useful in practice.

Level 0 is a special value: zstd treats it as a request for the default level, so compressing with level 0 is the same as compressing with level 3. `zstd_default_clevel()` itself never returns 0.

Value

An integer scalar.

Examples

```
zstd_min_clevel()
zstd_max_clevel()
zstd_default_clevel()
```

`zstd_tar_compress` *Compress files and directories into a Zstandard-compressed tar archive*

Description

Creates a `.tar.zst` archive: a tar archive of `files`, piped directly into the zstd streaming compressor without ever writing an uncompressed tar file to disk. Uses the streaming API, so memory use stays bounded regardless of the total size of `files`.

Usage

```
zstd_tar_compress(
  files,
  output,
  level = zstd_default_clevel(),
  dict = NULL,
  window_log = NULL,
  checksum = FALSE,
  strategy = NULL,
  nb_workers = 0L,
  content_size = TRUE,
  dict_id = TRUE,
  long_distance_matching = NULL
)
```

Arguments

- `files` — A character vector of paths to files and/or directories to archive. Directories are added recursively. Paths are stored in the archive as given, so use relative paths (e.g. after `setwd()` into the directory to archive) to avoid embedding local filesystem details in the archive.
- `output` — Path of the `.tar.zst` file to create. Overwritten if it already exists.
- `level` — Integer compression level. Defaults to `zstd_default_clevel()`, which is 3 on this system. Valid range is `zstd_min_clevel()` to `zstd_max_clevel()`, which is -131072 to 22 on this system.
- `dict` — `NULL`, or a raw vector containing a dictionary (as created by `zstd_train_dict()`, or any raw content dictionary). The same dictionary must be passed to `zstd_tar_decompress()`.
- `window_log` — `NULL` (library default), or an integer setting the maximum back-reference distance as a power of two, in bytes. Larger values can improve the compression ratio of large, redundant inputs, at the cost of memory use on both the compression and decompression side.
- `checksum` — Whether to store a checksum of the decompressed content in the frame. `zstd_mem_decompress()` and `zstd_decompress()` always verify this checksum automatically when present, and error if it does not match.
- `strategy` — `NULL` (library default for `level`), or one of `"fast"`, `"dfast"`, `"greedy"`, `"lazy"`, `"lazy2"`, `"btlazy2"`, `"btopt"`, `"btultra"`, `"btultra2"`, from fastest to strongest.
- `nb_workers` — Number of compression worker threads. `0` (the default) compresses on the calling thread. Higher values can speed up compression of large inputs at some cost to the compression ratio.
- `content_size` — Whether to store the decompressed size in the frame header, when known. Defaults to `TRUE`.
- `dict_id` — Whether to store the dictionary's ID in the frame header, when a dictionary is used. Defaults to `TRUE`.
- `long_distance_matching` — `NULL` (library default), or `TRUE`/ `FALSE` to force long distance matching on or off. Improves the compression ratio of large inputs with repetition far apart.

Details

Entries are stored using ustar-format tar headers. Relative paths up to about 254 bytes are supported (longer paths raise an error); files larger than 4GB are not supported.

Value

`output`, invisibly.

Examples

```
dir <- tempfile()
dir.create(file.path(dir, "subdir"), recursive = TRUE)
writeLines("hello", file.path(dir, "a.txt"))
writeLines("world", file.path(dir, "subdir", "b.txt"))
```

```
archive <- tempfile(fileext = ".tar.zst")
zstd_tar_compress(dir, archive)
exdir <- tempfile()
zstd_tar_decompress(archive, exdir)
readLines(file.path(exdir, basename(dir), "subdir", "b.txt"))
unlink(c(dir, archive, exdir), recursive = TRUE)
```

---

zstd_tar_decompress *Extract a Zstandard-compressed tar archive*

Description

Decompresses `input` to a temporary file and extracts it, using the streaming API for the decompression step, so memory use for that step stays bounded regardless of the total size of the archive.

Usage

```
zstd_tar_decompress(input, exdir, dict = NULL)
```

Arguments

`input`	Path to a `.tar.zst` file, as created by `zstd_tar_compress()`.
`exdir`	Directory to extract files into. Created if it doesn't already exist.
`dict`	`NULL`, or a raw vector containing the dictionary that was used to compress `input`. See `zstd_tar_compress()`.

Details

Plain ustar-format entries (as written by `zstd_tar_compress()`, with paths up to about 254 bytes) and PAX extended headers (as written by, e.g., macOS's default `tar`, for longer paths or link targets) are supported. `mtime`, and, outside Windows, `uid`/`gid` are applied to extracted files and directories (best-effort: e.g. `chown()` silently has no effect when not running as root). Archives created by other tools that use GNU long name/link headers are not supported, and raise an error rather than being extracted incorrectly.

Value

`exdir`, invisibly.

Examples

```
dir <- tempfile()
dir.create(dir)
writeLines("hello world", file.path(dir, "a.txt"))
archive <- tempfile(fileext = ".tar.zst")
zstd_tar_compress(dir, archive)
exdir <- tempfile()
```

```
zstd_tar_decompress(archive, exdir)
readLines(file.path(exdir, basename(dir), "a.txt"))
unlink(c(dir, archive, exdir), recursive = TRUE)
```

`zstd_train_dict` *Train a Zstandard dictionary from sample data*

Description

Dictionaries improve the compression ratio of many small, similar inputs, such as JSON records that share the same structure. Training needs a few hundred to a few thousand representative samples: aim for total sample size about 100x the target dictionary size.

Usage

```
zstd_train_dict(samples, size = 112640L)
```

Arguments

`samples`	A list of raw vectors, or a character vector of file paths (each file is one sample), representative of the data that will be compressed with the dictionary.
`size`	Target dictionary size, in bytes. Defaults to 112640 (110KB), the same default as the `zstd` command line tool.

Value

A raw vector: the trained dictionary. Pass it as the `dict` argument of `zstd_mem_compress()` and `zstd_mem_decompress()`.

Examples

```
samples <- lapply(1:100, function(i) {
  charToRaw(paste0('{"id":', i, ',"name":"sample"}'))
})
dict <- zstd_train_dict(samples, size = 1000)
cmp <- zstd_mem_compress(samples[[1]], dict = dict)
identical(zstd_mem_decompress(cmp, dict = dict), samples[[1]])
```

Index

zstd_compress, 2
zstd_compress(), *3*, *4*
zstd_decompress, 3
zstd_decompress(), *2*, *3*, *6*, *9*
zstd_default_clevel *(*zstd_min_clevel*)*, 7
zstd_default_clevel(), *2*, *6*, *9*
zstd_info, 4
zstd_max_clevel *(*zstd_min_clevel*)*, 7
zstd_max_clevel(), *2*, *6*, *9*
zstd_mem_compress, 5
zstd_mem_compress(), *7*, *11*
zstd_mem_decompress, 7
zstd_mem_decompress(), *3*, *6*, *9*, *11*
zstd_min_clevel, 7
zstd_min_clevel(), *2*, *6*, *9*
zstd_tar_compress, 8
zstd_tar_compress(), *10*
zstd_tar_decompress, 10
zstd_tar_decompress(), *9*
zstd_train_dict, 11
zstd_train_dict(), *2*, *6*, *9*