Package: zukomp (via r-universe)

October 16, 2026

Title Portable Byte Compression with a Pluggable Codec Registry

Version 0.1.0

Description Compresses and decompresses raw vectors through a single codec-neutral interface backed by a runtime registry, so the set of available codecs is discovered rather than fixed at compile time. The 'DEFLATE' family ('gzip', 'zlib', and headerless 'DEFLATE') is built in from vendored 'miniz' sources, requiring no system compression library. Decompression enforces configurable output-size and expansion-ratio limits and reports failures through structured conditions. A registered C-callable interface lets other packages drive the codecs from C and register codecs of their own. The formats are specified in Deutsch (1996) <doi:10.17487/RFC1951>, Deutsch and Gailly (1996) <doi:10.17487/RFC1950>, and Deutsch (1996) <doi:10.17487/RFC1952>.

License MIT + file LICENSE

URL https://github.com/pedrobtz/zukomp, https://pedrobtz.github.io/zukomp/

BugReports https://github.com/pedrobtz/zukomp/issues

Encoding UTF-8

Depends R (>= 4.0)

Imports utils

Suggests testthat (>= 3.2.0), withr

Config/Needs/website knitr, rmarkdown

Config/testthat/edition 3

Config/testthat/parallel true

Config/roxygen2/version 8.1.0

NeedsCompilation yes

Author Pedro Baltazar [aut, cre, cph], Rich Geldreich [ctb, cph] (author of the bundled miniz library), Tenacious Software LLC

[cph] (copyright holder of the bundled miniz library), RAD Game Tools [cph] (copyright holder of the bundled miniz library), Valve Software [cph] (copyright holder of the bundled miniz library), Martin Raiber [ctb, cph] (contributor to and copyright holder of the bundled miniz library), Alex Evans [ctb] (author of the PNG writer in the bundled miniz library), Alistair Moffat [ctb] (co-author of the code-length algorithm in the bundled miniz library), Jyrki Katajainen [ctb] (co-author of the code-length algorithm in the bundled miniz library)

Maintainer Pedro Baltazar <pedrobtz@gmail.com>

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

Date/Publication 2026-10-16 13:00:02 UTC

RemoteUrl https://github.com/cran/zukomp

RemoteRef HEAD

RemoteSha 4d6d28c23e8cf32b670d4e37ba8ceab57f8dc811

Contents

komp_codec_available 2
komp_codecs 3
komp_compress 4
komp_decompress 5
komp_detect 6
komp_info 7

Index **8**

`komp_codec_available` *Is a codec implementation available?*

Description

Is a codec implementation available?

Usage

```
komp_codec_available(codec)
```

Arguments

`codec`	A codec name, as listed in `komp_codecs()`'s `id` column.

Value

TRUE if an implementation is registered, FALSE if the codec is known to zukomp but not installed. An unknown name is an error of class zukomp_unsupported_codec, since it is far more likely to be a typo than a deliberate probe.

Examples

```
komp_codec_available("identity")

# a codec name zukomp reserves, with no implementation released yet
komp_codec_available("zstd")
```

---

komp_codecs *The codec capability table*

Description

Every codec this build of zukomp knows the name of, whether or not an implementation is present, plus any codec registered by another package. This is the R-visible face of the registry: it is how you discover what can be compressed, at what levels, and which HTTP content-coding each codec corresponds to.

Usage

```
komp_codecs()
```

Details

A codec that is declared but not installed still gets a row, with available = FALSE and NA for everything capability-shaped. That is deliberate: "zstd exists but you need the zukomp.zstd package" is a more useful answer than pretending the codec does not exist.

Value

A data frame with one row per codec and the columns:

id Codec name, as accepted by the codec argument elsewhere.

available Is an implementation registered?

can_encode, can_decode Supported directions; NA if unavailable.

level_min, level_max, level_default Codec-native compression levels. NA when the codec has no level axis, and when it is unavailable. Levels are not comparable between codecs.

level_fast, level_best Where the abstract levels "fast" and "best" land for this codec. These are *not* level_min and level_max: for the DEFLATE family level 0 is stored blocks, so "fast" is 1, and a codec whose level is an acceleration factor inverts the mapping entirely. Only the codec knows, so it declares them.

detectable Can `komp_detect()` recognise this codec from its bytes? Headerless formats cannot be detected and must be named.

can_flush Does the codec support a mid-stream flush – "put the bytes on the wire now"? `NA` if unavailable. A caller streaming a request body should check this before committing to a codec, rather than discovering it mid-body.

content_encoding The HTTP content-coding token, or `NA`.

source Package that registered the implementation, or `NA`.

References

The built-in codecs implement the formats specified in Deutsch, P. (1996) "DEFLATE Compressed Data Format Specification version 1.3", RFC 1951, doi:10.17487/RFC1951; Deutsch, P. and Gailly, J-L. (1996) "ZLIB Compressed Data Format Specification version 3.3", RFC 1950, doi:10.17487/RFC1950; and Deutsch, P. (1996) "GZIP file format specification version 4.3", RFC 1952, doi:10.17487/RFC1952.

Examples

```
codecs <- komp_codecs()
codecs[, c("id", "available", "content_encoding")]

# what this build can actually decompress right now
codecs$id[which(codecs$can_decode)]
```

`komp_compress` *Compress a raw vector*

Description

Bytes in, bytes out. Character input is not accepted: encoding is a decision the caller must make explicitly, so convert with `charToRaw()` or `iconv()` first.

Usage

```
komp_compress(x, codec = "gzip", level = NULL)
```

Arguments

`x`	A raw vector.
`codec`	Codec name, as listed in `komp_codecs()`. Defaults to `"gzip"`, which interoperates with everything.
`level`	Compression level. Either a codec-native whole number, one of the abstract names `"fast"`, `"default"` and `"best"`, or `NULL` for the codec's own default. Numeric levels are **not** comparable between codecs: `6` means different things to gzip and to zstd, and `komp_codecs()` publishes each codec's valid range. The abstract names are therefore the portable way to say "compress harder" – each codec declares where `"fast"` and `"best"` land (the `level_fast` and `level_best` columns), and they work on every codec, including ones with no level axis at all.

Value

A raw vector.

See Also

komp_decompress(), komp_codecs()

Examples

```
x <- charToRaw(strrep("compress me ", 100))
z <- komp_compress(x)
length(x)
length(z)
identical(komp_decompress(z, "gzip"), x)

# gzip output is deterministic: no timestamp, no filename
identical(komp_compress(x), komp_compress(x))

# abstract levels port across codecs; numeric ones do not
length(komp_compress(x, level = "fast"))
length(komp_compress(x, level = "best"))
```

---

komp_decompress *Decompress a raw vector*

Description

Decompress a raw vector

Usage

```
komp_decompress(
  x,
  codec = "auto",
  max_output = getOption("zukomp.max_output", 1024^3),
  max_ratio = getOption("zukomp.max_ratio", NULL)
)
```

Arguments

x	A raw vector holding a complete compressed stream.
codec	Codec name, as listed in komp_codecs().
max_output	Refuse to produce more than this many bytes. Defaults to 1 GiB, overridable with options(zukomp.max_output =). Use 0 for unlimited, deliberately. Compressed input from an untrusted source can expand enormously, and the cap is enforced by the core stream driver, so no codec can bypass it.
max_ratio	Refuse to expand by more than this factor. NULL (the default) means no ratio limit: legitimately compressible data routinely exceeds any safe-looking threshold, so this is opt-in.

Value

A raw vector.

See Also

komp_compress(), komp_codecs()

Examples

```
z <- komp_compress(charToRaw(strrep("data ", 200)), "gzip")
rawToChar(komp_decompress(z, "gzip"))

# a small input that would expand a long way is stopped, not allocated
bomb <- komp_compress(raw(1e6), "gzip")
length(bomb)
try(komp_decompress(bomb, "gzip", max_output = 1024))
```

---

komp_detect *Identify the codec that produced a compressed stream*

Description

Detection is a property of the codec registry, not a hardcoded list: a codec registered by another package becomes detectable as soon as it advertises magic bytes.

Usage

```
komp_detect(x)
```

Arguments

x	A raw vector; only the leading bytes are examined.

Details

Not every format can be detected. Raw DEFLATE, raw LZ4 blocks, raw Snappy and brotli have no header to recognise, so they return NA and must be named explicitly. This is deliberate: guessing wrong for a headerless format means silently returning the wrong bytes.

zlib is recognised by a header *predicate* rather than a constant, so it carries a small false-positive rate – roughly one arbitrary byte pair in a thousand looks like a valid zlib header. Codecs with real magic are always tested first.

Value

The codec's name, or NA_character_ if nothing matched.

See Also

komp_codecs(), whose detectable column says which codecs can be found this way.

Examples

```
komp_detect(komp_compress(charToRaw("hello"), "gzip"))
komp_detect(komp_compress(charToRaw("hello"), "zlib"))

# headerless formats are not guessed at
komp_detect(komp_compress(charToRaw("hello"), "deflate-raw"))
```

`komp_info` *Build and provenance information*

Description

What this build of zukomp actually contains: its version, the ABI version other packages link against, the codecs it registered, and the vendored sources compiled into it with the trim applied to them.

Usage

```
komp_info()
```

Details

Reported from the compiled library rather than read from `tools/vendor/manifest.tsv`, because the manifest is not installed and what matters here is what was actually built.

Value

A list with `version`, `abi_version`, `codecs`, `vendored` and `build_flags`. `version` is a character string rather than a `package_version` object: this is diagnostic output that mostly ends up pasted into a log line or a bug report, and `paste()` on a `package_version` needs an `as.character()` at every call site. Compare against `utils::packageVersion()` with `package_version(komp_info()$version)` if you need ordering.

Examples

```
info <- komp_info()
info$version
info$vendored
info$build_flags
```

Index

charToRaw(), *4*

komp_codec_available, 2
komp_codecs, 3
komp_codecs(), *2*, *4–6*
komp_compress, 4
komp_compress(), *6*
komp_decompress, 5
komp_decompress(), *5*
komp_detect, 6
komp_info, 7