Package 'b64'

March 31, 2025

Title Fast and Vectorized Base 64 Engine

Version 0.1.4

Description Provides a fast, lightweight, and vectorized base 64 engine to encode and decode character and raw vectors as well as files stored on disk. Common base 64 alphabets are supported out of the box including the standard, URL-safe, bcrypt, crypt, 'BinHex', and IMAP-modified UTF-7 alphabets. Custom engines can be created to support unique base 64 encoding and decoding needs.

License MIT + file LICENSE

Encoding UTF-8

Language en

RoxygenNote 7.3.2

Config/rextendr/version 0.3.1.9001

SystemRequirements Cargo (Rust's package manager), rustc

Suggests blob, testthat (>= 3.0.0)

Config/testthat/edition 3

URL https://extendr.github.io/b64/, https://github.com/extendr/b64

BugReports https://github.com/extendr/b64/issues

Depends R (>= 4.2)

NeedsCompilation yes

Author Josiah Parry [aut, cre] (<https://orcid.org/0000-0001-9910-865X>), Etienne Bacher [ctb] (<https://orcid.org/0000-0002-9271-5075>)

Maintainer Josiah Parry <josiah.parry@gmail.com>

Repository CRAN

Date/Publication 2025-03-31 16:10:06 UTC

Contents

alphabet 2
b64_chunk 3
encode 4
engine 5
new_config 6

Index **7**

`alphabet` *Standard base64 alphabets*

Description

Create an alphabet from a set of standard base64 alphabets, or use your own.

Usage

```
alphabet(which = "standard")

new_alphabet(chars)
```

Arguments

`which`	default `"standard"`. Which base64 alphabet to use. See details for other values.
`chars`	a character scalar contains 64 unique characters.

Details

- `"bcrypt"`: bcrypt alphabet
- `"bin_hex"`: alphabet used in BinHex 4.0 files
- `"crypt"`: crypt(3) alphabet (with . and / as the first two characters)
- `"imap_mutf7"`: alphabet used in IMAP-modified UTF-7 (with + and ,)
- `"standard"`: standard alphabet (with + and /) specified in RFC 4648
- `"url_safe"`: URL-safe alphabet (with - and _) specified in RFC 4648

See base64 crate from where these definitions come.

Value

an object of class `alphabet`

Examples

```
alphabet("standard")
alphabet("bcrypt")
alphabet("bin_hex")
alphabet("crypt")
alphabet("imap_mutf7")
alphabet("url_safe")

new_alphabet("qwertyuiop[]asdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM1234567890")
```

---

b64_chunk *Utility Functions*

Description

Functions to perform common tasks when working with base64 encoded strings.

Usage

```
b64_chunk(encoded, width)

b64_wrap(chunks, newline)
```

Arguments

encoded	a character vector of base64 encoded strings.
width	a numeric scalar defining the width of the chunks. Must be divisible by 4.
chunks	a character vector of base64 encoded strings.
newline	a character scalar defining the newline character.

Details

`b64_chunk()` splits a character vector of base64 encoded strings into chunks of a specified width.

`b64_wrap()` wraps a character vector of base64 encoded strings with a newline character.

Value

- `b64_chunk()` returns a list of character vectors.
- `b64_wrap()` returns a scalar character vector.

Examples

```
encoded <- encode("Hello, world!")
chunked <- b64_chunk(encoded, 4)
chunked

b64_wrap(chunked, "\n")
```

encode *Encode and decode using base64*

Description

Encode and decode using base64

Usage

```
encode(what, eng = engine())

decode(what, eng = engine())

encode_file(path, eng = engine())

decode_file(path, eng = engine())
```

Arguments

what	a character, raw, or blob vector
eng	a base64 engine. See `engine()` for details.
path	a path to a base64 encoded file.

Value

Both `encode()` and `decode()` are vectorized. They will return a character and blob vector the same length as `what`, respectively.

Examples

```
# encode hello world
encoded <- encode("Hello world")
encoded

# decode to a blob
decoded <- decode(encoded)
decoded

# convert back to a character
rawToChar(decoded[[1]])
```

`engine` *Create an encoding engine*

Description

Create an encoding engine

Usage

```
engine(which = "standard")

new_engine(.alphabet = alphabet(), .config = new_config())
```

Arguments

`which`	default `"standard"`. The base64 encoding engine to be used. See details for more.
`.alphabet`	an object of class `alphabet` as created with `alphabet()` or `new_alphabet()`
`.config`	an object of class `engine_config` as created with `new_config()`

Details

Engines:

By default, the "standard" base64 engine is used which is specified in RFC 4648.

Additional pre-configured base64 engines are provided these are:

- `"standard_no_pad"`: uses the standard engine without padding
- `"url_safe"`: uses a url-safe alphabet with padding
- `"url_safe_no_pad"`: uses a url-safe alphabet without padding

See base64 crate for more.

Value

an object of class `engine`.

Examples

```
engine()
new_engine(alphabet("bcrypt"), new_config())
```

`new_config` *Create a custom encoding engine*

Description

Create a custom encoding engine

Usage

```
new_config(
  encode_padding = TRUE,
  decode_padding_trailing_bits = FALSE,
  decode_padding_mode = c("canonical", "indifferent", "none")
)
```

Arguments

`encode_padding`	default `TRUE` add 1-2 trailing = to pad results
`decode_padding_trailing_bits`	default `FALSE`. "If invalid trailing bits are present and this is true, those bits will be silently ignored." (See details for reference).
`decode_padding_mode`	default `"canonical"`. Other values are `"indifferent"` and `"none"`. See details for more.

Details

See base64 crate for more details.

Decode Padding Modes:

There are three modes that can be used for `decode_padding_mode` argument.

- `"canonical"`: padding must consist of 0, 1, or 2 = characters
- `"none"`: there must be no padding characters present
- `"indifferent"`: canonical padding is used, but omitted padding characters are also permitted

Value

an object of class `engine_config`

Examples

```
# create a new nonsensicle config
new_config(FALSE, TRUE, "none")
```

Index

alphabet, 2
alphabet(), *5*

b64_chunk, 3
b64_wrap *(*b64_chunk*)*, 3

decode *(*encode*)*, 4
decode_file *(*encode*)*, 4

encode, 4
encode_file *(*encode*)*, 4
engine, 5
engine(), *4*

new_alphabet *(*alphabet*)*, 2
new_alphabet(), *5*
new_config, 6
new_config(), *5*
new_engine *(*engine*)*, 5