

Discover Packages > Standard library > encoding > base64 

# base64





package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: 3 |

Imported by: [122,787](#)

## Details

- ✓ Valid [go.mod](#) file 
- ✓ Redistributable license 
- ✓ Tagged version 
- ✓ Stable version 

[Learn more](#)

## Repository

[cs.opensource.google/go/go](https://cs.opensource.google/go/go)

## Links

 [Report a Vulnerability](#)

 Documentation 

## <> Documentation

### Overview

Package base64 implements base64 encoding as specified by [RFC 4648](#).

► [Example](#)

### Index

[Constants](#)

[Variables](#)

[func NewDecoder\(enc \\*Encoding, r io.Reader\) io.Reader](#)

[func NewEncoder\(enc \\*Encoding, w io.Writer\) io.WriteCloser](#)

[type CorruptInputError](#)

[func \(e CorruptInputError\) Error\(\) string](#)

[type Encoding](#)

[func NewEncoding\(encoder string\) \\*Encoding](#)

[func \(enc \\*Encoding\) Decode\(dst, src \[\]byte\) \(n int, err error\)](#)

[func \(enc \\*Encoding\) DecodeString\(s string\) \(\[\]byte, error\)](#)

[func \(enc \\*Encoding\) DecodedLen\(n int\) int](#)

[func \(enc \\*Encoding\) Encode\(dst, src \[\]byte\)](#)

[func \(enc \\*Encoding\) EncodeToString\(src \[\]byte\) string](#)

[func \(enc \\*Encoding\) EncodedLen\(n int\) int](#)

[func \(enc Encoding\) Strict\(\) \\*Encoding](#)

[func \(enc Encoding\) WithPadding\(padding rune\) \\*Encoding](#)

## Examples

[Package](#)

[Encoding.Decode](#)

[Encoding.DecodeString](#)

[Encoding.Encode](#)

[Encoding.EncodeToString](#)

[NewEncoder](#)

## Constants

[View Source](#)

```
const (  
    StdPadding rune = '=' // Standard padding character  
    NoPadding  rune = -1  // No padding  
)
```

## Variables

[View Source](#)

```
var RawStdEncoding = StdEncoding.WithPadding(NoPadding)
```

RawStdEncoding is the standard raw, unpadded base64 encoding, as defined in [RFC 4648 section 3.2](#). This is the same as StdEncoding but omits padding characters.

[View Source](#)

```
var RawURLEncoding = URLEncoding.WithPadding(NoPadding)
```

RawURLEncoding is the unpadded alternate base64 encoding defined in [RFC 4648](#). It is typically used in URLs and file names. This is the same as URLEncoding but omits padding characters.

[View Source](#)

```
var StdEncoding = NewEncoding(encodeStd)
```

StdEncoding is the standard base64 encoding, as defined in [RFC 4648](#).

[View Source](#)

```
var URLEncoding = NewEncoding(encodeURL)
```

URLEncoding is the alternate base64 encoding defined in [RFC 4648](#). It is typically used in URLs and file names.

## Functions

**func** [NewDecoder](#)

```
func NewDecoder(enc *Encoding, r io.Reader) io.Reader
```

NewDecoder constructs a new base64 stream decoder.

## func NewEncoder

```
func NewEncoder(enc *Encoding, w io.Writer) io.WriteCloser
```

NewEncoder returns a new base64 stream encoder. Data written to the returned writer will be encoded using enc and then written to w. Base64 encodings operate in 4-byte blocks; when finished writing, the caller must Close the returned encoder to flush any partially written blocks.

► [Example](#)

## Types

### type CorruptInputError

```
type CorruptInputError int64
```

### func (CorruptInputError) Error

```
func (e CorruptInputError) Error() string
```

### type Encoding

```
type Encoding struct {  
    // contains filtered or unexported fields  
}
```

An Encoding is a radix 64 encoding/decoding scheme, defined by a 64-character alphabet. The most common encoding is the "base64" encoding defined in [RFC 4648](#) and used in MIME ([RFC 2045](#)) and PEM ([RFC 1421](#)). [RFC 4648](#) also defines an alternate encoding, which is the standard encoding with - and \_ substituted for + and /.

### func NewEncoding

```
func NewEncoding(encoder string) *Encoding
```

NewEncoding returns a new padded Encoding defined by the given alphabet, which must be a 64-byte string that does not contain the padding character or CR / LF ('\r', '\n'). The resulting Encoding uses the default padding character ('='), which may be changed or disabled via WithPadding.

### func (\*Encoding) Decode

```
func (enc *Encoding) Decode(dst, src []byte) (n int, err error)
```

Decode decodes src using the encoding enc. It writes at most DecodedLen(len(src)) bytes to dst and returns the number of bytes written. If src contains invalid base64 data, it will return the number of bytes successfully written and CorruptInputError. New line characters (\r and \n) are ignored.

► [Example](#)

### func (\*Encoding) DecodeString

```
func (enc *Encoding) DecodeString(s string) ([]byte, error)
```

DecodeString returns the bytes represented by the base64 string s.

► [Example](#)

### func (\*Encoding) DecodedLen

```
func (enc *Encoding) DecodedLen(n int) int
```

DecodedLen returns the maximum length in bytes of the decoded data corresponding to n bytes of base64-encoded data.

### func (\*Encoding) Encode

```
func (enc *Encoding) Encode(dst, src []byte)
```

Encode encodes src using the encoding enc, writing EncodedLen(len(src)) bytes to dst.

The encoding pads the output to a multiple of 4 bytes, so Encode is not appropriate for use on individual blocks of a large data stream. Use NewEncoder() instead.

► [Example](#)

### func (\*Encoding) EncodeToString

```
func (enc *Encoding) EncodeToString(src []byte) string
```

EncodeToString returns the base64 encoding of src.

► [Example](#)

### func (\*Encoding) EncodedLen

```
func (enc *Encoding) EncodedLen(n int) int
```

EncodedLen returns the length in bytes of the base64 encoding of an input buffer of length n.

## func (Encoding) Strict

added in go1.8

```
func (enc Encoding) Strict() *Encoding
```

Strict creates a new encoding identical to enc except with strict decoding enabled. In this mode, the decoder requires that trailing padding bits are zero, as described in [RFC 4648 section 3.5](#).

Note that the input is still malleable, as new line characters (CR and LF) are still ignored.

## func (Encoding) WithPadding

added in go1.5

```
func (enc Encoding) WithPadding(padding rune) *Encoding
```

WithPadding creates a new encoding identical to enc except with a specified padding character, or NoPadding to disable padding. The padding character must not be '\r' or '\n', must not be contained in the encoding's alphabet and must be a rune equal or below '\xff'.



## Source Files

[View all](#) 

[base64.go](#)

### Why Go

[Use Cases](#)

[Case Studies](#)

### Get Started

[Playground](#)

[Tour](#)

[Stack Overflow](#)

[Help](#)

### Packages

[Standard Library](#)

[About Go Packages](#)

### About

[Download](#)

[Blog](#)

[Issue Tracker](#)

[Release Notes](#)

[Brand Guidelines](#)

[Code of Conduct](#)

## Connect

[Twitter](#)

[GitHub](#)

[Slack](#)

[r/golang](#)

[Meetup](#)

[Golang Weekly](#)

---

[Copyright](#)

[Terms of Service](#)

[Privacy Policy](#)

[Report an Issue](#)



Google