base32

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 2 |
Imported by: 5,248

Details Valid go.mod file ? Redistributable license ? Tagged version ?
Stable version ?
Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Documentation

Overview

Package base32 implements base32 encoding as specified by RFC 4648.

Index

Examples

Constants

View Source

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

)
```

Variables

View Source

```
var HexEncoding = NewEncoding(encodeHex)
```

HexEncoding is the “Extended Hex Alphabet” defined in RFC 4648. It is typically used in DNS.

View Source

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

StdEncoding is the standard base32 encoding, as defined in RFC 4648.

Functions

func NewDecoder

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

NewDecoder constructs a new base32 stream decoder.

func NewEncoder

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

NewEncoder returns a new base32 stream encoder. Data written to the returned writer will be encoded using enc and then written to w. Base32 encodings operate in 5-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 32 encoding/decoding scheme, defined by a 32-character alphabet. The most common is the "base32" encoding introduced for SASL GSSAPI and standardized in RFC 4648. The alternate "base32hex" encoding is used in DNSSEC.

func NewEncoding

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

NewEncoding returns a new Encoding defined by the given alphabet, which must be a 32-byte string.

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 base32 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 base32 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 base32-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 8 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 base32 encoding of src.

▶ Example

func (*Encoding) EncodedLen

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

EncodedLen returns the length in bytes of the base32 encoding of an input buffer of length n.

func (Encoding) WithPadding

added in go1.9

```
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

base32.go