

[Discover Packages](#) > [Standard library](#) > [mime](#) 



mime

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: [12](#) |Imported by: [17,774](#)



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

Rendered for [linux/amd64](#) 

Overview

Package mime implements parts of the MIME spec.

Index

[Constants](#)[Variables](#)[func AddExtensionType\(ext, typ string\) error](#)[func ExtensionsByType\(typ string\) \(\[\]string, error\)](#)[func FormatMediaType\(t string, param map\[string\]string\) string](#)[func ParseMediaType\(v string\) \(mediatype string, params map\[string\]string, err error\)](#)[func TypeByExtension\(ext string\) string](#)[type WordDecoder](#)[func \(d *WordDecoder\) Decode\(word string\) \(string, error\)](#)[func \(d *WordDecoder\) DecodeHeader\(header string\) \(string, error\)](#)[type WordEncoder](#)[func \(e WordEncoder\) Encode\(charset, s string\) string](#)

Examples

[FormatMediaType](#)[ParseMediaType](#)[WordDecoder.Decode](#)[WordDecoder.DecodeHeader](#)[WordEncoder.Encode](#)

Constants

[View Source](#)

```
const (  
    // BEncoding represents Base64 encoding scheme as defined by RFC 2045.  
    BEncoding = WordEncoder('b')  
    // QEncoding represents the Q-encoding scheme as defined by RFC 2047.  
    QEncoding = WordEncoder('q')  
)
```

Variables

[View Source](#)

```
var ErrInvalidMediaParameter = errors.New("mime: invalid media parameter")
```

ErrInvalidMediaParameter is returned by ParseMediaType if the media type value was found but there was an error parsing the optional parameters

Functions

func AddExtensionType

```
func AddExtensionType(ext, typ string) error
```

AddExtensionType sets the MIME type associated with the extension ext to typ. The extension should begin with a leading dot, as in ".html".

func ExtensionsByType

added in go1.5

```
func ExtensionsByType(typ string) ([]string, error)
```

ExtensionsByType returns the extensions known to be associated with the MIME type typ. The returned extensions will each begin with a leading dot, as in ".html". When typ has no associated extensions, ExtensionsByType returns an nil slice.

func FormatMediaType

```
func FormatMediaType(t string, param map[string]string) string
```

FormatMediaType serializes mediatype t and the parameters param as a media type conforming to RFC 2045 and RFC 2616. The type and parameter names are written in lower-case. When any of the arguments result in a standard violation then FormatMediaType returns the empty string.

► [Example](#)

func ParseMediaType

```
func ParseMediaType(v string) (mediatype string, params map[string]string, err error)
```

`ParseMediaType` parses a media type value and any optional parameters, per [RFC 1521](#). Media types are the values in Content-Type and Content-Disposition headers ([RFC 2183](#)). On success, `ParseMediaType` returns the media type converted to lowercase and trimmed of white space and a non-nil map. If there is an error parsing the optional parameter, the media type will be returned along with the error `ErrInvalidMediaParameter`. The returned map, `params`, maps from the lowercase attribute to the attribute value with its case preserved.

► Example

func `TypeByExtension`

```
func TypeByExtension(ext string) string
```

`TypeByExtension` returns the MIME type associated with the file extension `ext`. The extension `ext` should begin with a leading dot, as in `".html"`. When `ext` has no associated type, `TypeByExtension` returns `""`.

Extensions are looked up first case-sensitively, then case-insensitively.

The built-in table is small but on unix it is augmented by the local system's MIME-info database or `mime.types` file(s) if available under one or more of these names:

```
/usr/local/share/mime/globs2
/usr/share/mime/globs2
/etc/mime.types
/etc/apache2/mime.types
/etc/apache/mime.types
```

On Windows, MIME types are extracted from the registry.

Text types have the charset parameter set to `"utf-8"` by default.

Types

type `WordDecoder`

added in go1.5

```
type WordDecoder struct {
    // CharsetReader, if non-nil, defines a function to generate
    // charset-conversion readers, converting from the provided
    // charset into UTF-8.
    // Charsets are always lower-case. utf-8, iso-8859-1 and us-ascii charsets
    // are handled by default.
    // One of the CharsetReader's result values must be non-nil.
    CharsetReader func(charset string, input io.Reader) (io.Reader, error)
}
```

A `WordDecoder` decodes MIME headers containing [RFC 2047](#) encoded-words.

func (*WordDecoder) Decode

added in go1.5

```
func (d *WordDecoder) Decode(word string) (string, error)
```

Decode decodes an [RFC 2047](#) encoded-word.

► [Example](#)

func (*WordDecoder) DecodeHeader

added in go1.5

```
func (d *WordDecoder) DecodeHeader(header string) (string, error)
```

DecodeHeader decodes all encoded-words of the given string. It returns an error if and only if CharsetReader of d returns an error.

► [Example](#)

type WordEncoder

added in go1.5

```
type WordEncoder byte
```

A WordEncoder is an [RFC 2047](#) encoded-word encoder.

func (WordEncoder) Encode

added in go1.5

```
func (e WordEncoder) Encode(charset, s string) string
```

Encode returns the encoded-word form of s. If s is ASCII without special characters, it is returned unchanged. The provided charset is the IANA charset name of s. It is case insensitive.

► [Example](#)



Source Files

[View all](#)

[encodedword.go](#)
[grammar.go](#)

[mediatype.go](#)
[type.go](#)

[type_unix.go](#)



Directories

[multipart](#)

Package multipart implements MIME multipart parsing, as defined in RFC 2046.

[quotedprintable](#)

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