

[Discover Packages](#) > [Standard library](#) > [net](#) > [http](#) > [httputil](#) 





httputil

package

standard library

Version: [go1.20.1](#) **Latest** | Published: Feb 14, 2023 | License: [BSD-3-Clause](#) | Imports: [19](#) |Imported by: [18,734](#)

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 `httputil` provides HTTP utility functions, complementing the more common ones in the `net/http` package.

Index

Variables

```
func DumpRequest(req *http.Request, body bool) ([]byte, error)
func DumpRequestOut(req *http.Request, body bool) ([]byte, error)
func DumpResponse(resp *http.Response, body bool) ([]byte, error)
func NewChunkedReader(r io.Reader) io.Reader
func NewChunkedWriter(w io.Writer) io.WriteCloser
type BufferPool
type ClientConn DEPRECATED
    func NewClientConn(c net.Conn, r *bufio.Reader) *ClientConn DEPRECATED
    func NewProxyClientConn(c net.Conn, r *bufio.Reader) *ClientConn DEPRECATED
    func (cc *ClientConn) Close() error
    func (cc *ClientConn) Do(req *http.Request) (*http.Response, error)
    func (cc *ClientConn) Hijack() (c net.Conn, r *bufio.Reader)
    func (cc *ClientConn) Pending() int
    func (cc *ClientConn) Read(req *http.Request) (resp *http.Response, err error)
    func (cc *ClientConn) Write(req *http.Request) error
type ProxyRequest
    func (r *ProxyRequest) SetURL(target *url.URL)
```

```
func (r *ProxyRequest) SetXForwarded()
type ReverseProxy
func NewSingleHostReverseProxy(target *url.URL) *ReverseProxy
func (p *ReverseProxy) ServeHTTP(rw http.ResponseWriter, req *http.Request)
type ServerConn DEPRECATED
func NewServerConn(c net.Conn, r *bufio.Reader) *ServerConn DEPRECATED
func (sc *ServerConn) Close() error
func (sc *ServerConn) Hijack() (net.Conn, *bufio.Reader)
func (sc *ServerConn) Pending() int
func (sc *ServerConn) Read() (*http.Request, error)
func (sc *ServerConn) Write(req *http.Request, resp *http.Response) error
```

Examples

[DumpRequest](#)

[DumpRequestOut](#)

[DumpResponse](#)

[ReverseProxy](#)

Constants

This section is empty.

Variables

[View Source](#)

```
var (
    // Deprecated: No longer used.
    ErrPersistEOF = &http.ProtocolError{ErrorString: "persistent connection closed"}

    // Deprecated: No longer used.
    ErrClosed = &http.ProtocolError{ErrorString: "connection closed by user"}

    // Deprecated: No longer used.
    ErrPipeline = &http.ProtocolError{ErrorString: "pipeline error"}
)
```

[View Source](#)

```
var ErrLineTooLong = internal.ErrLineTooLong
```

ErrLineTooLong is returned when reading malformed chunked data with lines that are too long.

Functions

func [DumpRequest](#)

```
func DumpRequest(req *http.Request, body bool) ([]byte, error)
```

DumpRequest returns the given request in its HTTP/1.x wire representation. It should only be used by servers to debug client requests. The returned representation is an approximation only; some details of the initial request are lost while parsing it into an `http.Request`. In particular, the order and case of header field names are lost. The order of values in multi-valued headers is kept intact. HTTP/2 requests are dumped in HTTP/1.x form, not in their original binary representations.

If `body` is true, DumpRequest also returns the body. To do so, it consumes `req.Body` and then replaces it with a new `io.ReadCloser` that yields the same bytes. If DumpRequest returns an error, the state of `req` is undefined.

The documentation for `http.Request.Write` details which fields of `req` are included in the dump.

► [Example](#)

func DumpRequestOut

```
func DumpRequestOut(req *http.Request, body bool) ([]byte, error)
```

DumpRequestOut is like DumpRequest but for outgoing client requests. It includes any headers that the standard `http.Transport` adds, such as `User-Agent`.

► [Example](#)

func DumpResponse

```
func DumpResponse(resp *http.Response, body bool) ([]byte, error)
```

DumpResponse is like DumpRequest but dumps a response.

► [Example](#)

func NewChunkedReader

```
func NewChunkedReader(r io.Reader) io.Reader
```

NewChunkedReader returns a new chunkedReader that translates the data read from `r` out of HTTP "chunked" format before returning it. The chunkedReader returns `io.EOF` when the final 0-length chunk is read.

NewChunkedReader is not needed by normal applications. The `http` package automatically decodes chunking when reading response bodies.

func NewChunkedWriter

```
func NewChunkedWriter(w io.Writer) io.WriteCloser
```

NewChunkedWriter returns a new chunkedWriter that translates writes into HTTP "chunked" format before writing them to w. Closing the returned chunkedWriter sends the final 0-length chunk that marks the end of the stream but does not send the final CRLF that appears after trailers; trailers and the last CRLF must be written separately.

NewChunkedWriter is not needed by normal applications. The http package adds chunking automatically if handlers don't set a Content-Length header. Using NewChunkedWriter inside a handler would result in double chunking or chunking with a Content-Length length, both of which are wrong.

Types

type **BufferPool**

added in go1.6

```
type BufferPool interface {  
    Get() []byte  
    Put([]byte)  
}
```

A BufferPool is an interface for getting and returning temporary byte slices for use by io.CopyBuffer.

type **ClientConn** DEPRECATED [Show](#)

type **ProxyRequest**

added in go1.20

```
type ProxyRequest struct {  
    // In is the request received by the proxy.  
    // The Rewrite function must not modify In.  
    In *http.Request  
  
    // Out is the request which will be sent by the proxy.  
    // The Rewrite function may modify or replace this request.  
    // Hop-by-hop headers are removed from this request  
    // before Rewrite is called.  
    Out *http.Request  
}
```

A ProxyRequest contains a request to be rewritten by a ReverseProxy.

func (*ProxyRequest) **SetURL**

added in go1.20

```
func (r *ProxyRequest) SetURL(target *url.URL)
```

SetURL routes the outbound request to the scheme, host, and base path provided in target. If the target's path is "/base" and the incoming request was for "/dir", the target request will be for "/base/dir".

SetURL rewrites the outbound Host header to match the target's host. To preserve the inbound request's Host header (the default behavior of NewSingleHostReverseProxy):

```
rewriteFunc := func(r *httputil.ProxyRequest) {  
    r.SetURL(url)  
    r.Out.Host = r.In.Host  
}
```

func (*ProxyRequest) SetXForwarded

added in go1.20

```
func (r *ProxyRequest) SetXForwarded()
```

SetXForwarded sets the X-Forwarded-For, X-Forwarded-Host, and X-Forwarded-Proto headers of the outbound request.

- The X-Forwarded-For header is set to the client IP address.
- The X-Forwarded-Host header is set to the host name requested by the client.
- The X-Forwarded-Proto header is set to "http" or "https", depending on whether the inbound request was made on a TLS-enabled connection.

If the outbound request contains an existing X-Forwarded-For header, SetXForwarded appends the client IP address to it. To append to the inbound request's X-Forwarded-For header (the default behavior of ReverseProxy when using a Director function), copy the header from the inbound request before calling SetXForwarded:

```
rewriteFunc := func(r *httputil.ProxyRequest) {  
    r.Out.Header["X-Forwarded-For"] = r.In.Header["X-Forwarded-For"]  
    r.SetXForwarded()  
}
```

type ReverseProxy

```
type ReverseProxy struct {  
    // Rewrite must be a function which modifies  
    // the request into a new request to be sent  
    // using Transport. Its response is then copied  
    // back to the original client unmodified.  
    // Rewrite must not access the provided ProxyRequest  
    // or its contents after returning.  
    //  
    // The Forwarded, X-Forwarded, X-Forwarded-Host,  
    // and X-Forwarded-Proto headers are removed from the  
    // outbound request before Rewrite is called. See also  
    // the ProxyRequest.SetXForwarded method.  
    //  
    // Unparsable query parameters are removed from the  
    // outbound request before Rewrite is called.  
    // The Rewrite function may copy the inbound URL's  
    // RawQuery to the outbound URL to preserve the original  
    // parameter string. Note that this can lead to security  
    // issues if the proxy's interpretation of query parameters  
    // does not match that of the downstream server.  
    //
```

```

// At most one of Rewrite or Director may be set.
Rewrite func(*ProxyRequest)

// Director is a function which modifies
// the request into a new request to be sent
// using Transport. Its response is then copied
// back to the original client unmodified.
// Director must not access the provided Request
// after returning.
//
// By default, the X-Forwarded-For header is set to the
// value of the client IP address. If an X-Forwarded-For
// header already exists, the client IP is appended to the
// existing values. As a special case, if the header
// exists in the Request.Header map but has a nil value
// (such as when set by the Director func), the X-Forwarded-For
// header is not modified.
//
// To prevent IP spoofing, be sure to delete any pre-existing
// X-Forwarded-For header coming from the client or
// an untrusted proxy.
//
// Hop-by-hop headers are removed from the request after
// Director returns, which can remove headers added by
// Director. Use a Rewrite function instead to ensure
// modifications to the request are preserved.
//
// Unparsable query parameters are removed from the outbound
// request if Request.Form is set after Director returns.
//
// At most one of Rewrite or Director may be set.
Director func(*http.Request)

// The transport used to perform proxy requests.
// If nil, http.DefaultTransport is used.
Transport http.RoundTripper

// FlushInterval specifies the flush interval
// to flush to the client while copying the
// response body.
// If zero, no periodic flushing is done.
// A negative value means to flush immediately
// after each write to the client.
// The FlushInterval is ignored when ReverseProxy
// recognizes a response as a streaming response, or
// if its ContentLength is -1; for such responses, writes
// are flushed to the client immediately.
FlushInterval time.Duration

// ErrorLog specifies an optional logger for errors
// that occur when attempting to proxy the request.
// If nil, logging is done via the log package's standard logger.
ErrorLog *log.Logger

```

```

// BufferPool optionally specifies a buffer pool to
// get byte slices for use by io.CopyBuffer when
// copying HTTP response bodies.
BufferPool BufferPool

// ModifyResponse is an optional function that modifies the
// Response from the backend. It is called if the backend
// returns a response at all, with any HTTP status code.
// If the backend is unreachable, the optional ErrorHandler is
// called without any call to ModifyResponse.
//
// If ModifyResponse returns an error, ErrorHandler is called
// with its error value. If ErrorHandler is nil, its default
// implementation is used.
ModifyResponse func(*http.Response) error

// ErrorHandler is an optional function that handles errors
// reaching the backend or errors from ModifyResponse.
//
// If nil, the default is to log the provided error and return
// a 502 Status Bad Gateway response.
ErrorHandler func(http.ResponseWriter, *http.Request, error)
}

```

ReverseProxy is an HTTP Handler that takes an incoming request and sends it to another server, proxying the response back to the client.

1xx responses are forwarded to the client if the underlying transport supports `ClientTrace.Got1xxResponse`.

► Example

func NewSingleHostReverseProxy

```
func NewSingleHostReverseProxy(target *url.URL) *ReverseProxy
```

`NewSingleHostReverseProxy` returns a new `ReverseProxy` that routes URLs to the scheme, host, and base path provided in `target`. If the target's path is `"/base"` and the incoming request was for `"/dir"`, the target request will be for `/base/dir`.

`NewSingleHostReverseProxy` does not rewrite the `Host` header.

To customize the `ReverseProxy` behavior beyond what `NewSingleHostReverseProxy` provides, use `ReverseProxy` directly with a `Rewrite` function. The `ProxyRequest SetURL` method may be used to route the outbound request. (Note that `SetURL`, unlike `NewSingleHostReverseProxy`, rewrites the `Host` header of the outbound request by default.)

```

proxy := &ReverseProxy{
    Rewrite: func(r *ProxyRequest) {

```

```
    r.SetURL(target)
    r.Out.Host = r.In.Host // if desired
}
}
```

func (*ReverseProxy) ServeHTTP

```
func (p *ReverseProxy) ServeHTTP(rw http.ResponseWriter, req *http.Request)
```

type ServerConn DEPRECATED [Show](#)

Source Files

[View all](#) 

[dump.go](#)
[httputil.go](#)

[persist.go](#)
[reverseproxy.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