net / smtp

Package smtp go1.15.2 Latest

Published: **Sep 9, 2020** | License: BSD-3-Clause | Standard library

Doc Overview Subdirectories Versions Imports Imported By Licenses

Overview

Package smtp implements the Simple Mail Transfer Protocol as defined in RFC 5321. It also implements the following extensions:

```
8BITMIME  RFC 1652
AUTH      RFC 2554
STARTTLS  RFC 3207
```

Additional extensions may be handled by clients.

The smtp package is frozen and is not accepting new features. Some external packages provide more functionality. See:

```
https://godoc.org/?q=smtp
```

func SendMail

```
func SendMail(addr string, a Auth, from string, to []string, msg []byte) error
```

SendMail connects to the server at addr, switches to TLS if possible, authenticates with the optional mechanism a if possible, and then sends an email from address from, to addresses to, with message msg. The addr must include a port, as in "mail.example.com:smtp".

The addresses in the to parameter are the SMTP RCPT addresses.

The msg parameter should be an RFC 822-style email with headers first, a blank line, and then the message body. The lines of msg should be CRLF terminated. The msg headers should usually include fields such as "From", "To", "Subject", and "Cc". Sending "Bcc" messages is accomplished by including an email address in the to parameter but not including it in the msg headers.

The SendMail function and the net/smtp package are low-level mechanisms and provide no support for DKIM signing, MIME attachments (see the mime/multipart package), or other mail functionality. Higher-level packages exist outside of the standard library.

type Auth

```
type Auth interface {
    // Start begins an authentication with a server.
    // It returns the name of the authentication protocol
    // and optionally data to include in the initial AUTH message
    // sent to the server. It can return proto == "" to indicate
    // that the authentication should be skipped.
    // If it returns a non-nil error, the SMTP client aborts
    // the authentication attempt and closes the connection.
    Start(server *ServerInfo) (proto string, toServer []byte, err error)

    // Next continues the authentication. The server has just sent
    // the fromServer data. If more is true, the server expects a
    // response, which Next should return as toServer; otherwise
    // Next should return toServer == nil.
    // If Next returns a non-nil error, the SMTP client aborts
    // the authentication attempt and closes the connection.
    Next(fromServer []byte, more bool) (toServer []byte, err error)
}
```

Auth is implemented by an SMTP authentication mechanism.

func CRAMMD5Auth

```
func CRAMMD5Auth(username, secret string) Auth
```

CRAMMD5Auth returns an Auth that implements the CRAM-MD5 authentication mechanism as defined in RFC 2195. The returned Auth uses the given username and secret to authenticate to the server using the challenge-response mechanism.

func PlainAuth

```
func PlainAuth(identity, username, password, host string) Auth
```

PlainAuth returns an Auth that implements the PLAIN authentication mechanism as defined in RFC 4616. The returned Auth uses the given username and password to authenticate to host and act as identity. Usually identity should be the empty string, to act as username.

PlainAuth will only send the credentials if the connection is using TLS or is connected to localhost. Otherwise authentication will fail with an error, without sending the credentials.

type Client

```
type Client struct {
    // Text is the textproto.Conn used by the Client. It is exported to allow for
    // clients to add extensions.
    Text *textproto.Conn
    // contains filtered or unexported fields
}
```

A Client represents a client connection to an SMTP server.

func Dial

```
func Dial(addr string) (*Client, error)
```

Dial returns a new Client connected to an SMTP server at addr. The addr must include a port, as in "mail.example.com:smtp".

func NewClient

```
func NewClient(conn net.Conn, host string) (*Client, error)
```

NewClient returns a new Client using an existing connection and host as a server name to be used when authenticating.

func (*Client) Auth

```
func (c *Client) Auth(a Auth) error
```

Auth authenticates a client using the provided authentication mechanism. A failed authentication closes the connection. Only servers that advertise the AUTH extension support this function.

func (*Client) Close

```
func (c *Client) Close() error
```

Close closes the connection.

func (*Client) Data

```
func (c *Client) Data() (io.WriteCloser, error)
```

Data issues a DATA command to the server and returns a writer that can be used to write the mail headers and body. The caller should close the writer before calling any more methods on c. A call to Data must be preceded by one or more calls to Rcpt.

func (*Client) Extension

```
func (c *Client) Extension(ext string) (bool, string)
```

Extension reports whether an extension is support by the server. The extension name is case-insensitive. If the extension is supported, Extension also returns a string that contains any parameters the server specifies for the extension.

func (*Client) Hello

```
func (c *Client) Hello(localName string) error
```

Hello sends a HELO or EHLO to the server as the given host name. Calling this method is only necessary if the client needs control over the host name used. The client will introduce itself as "localhost" automatically otherwise. If Hello is called, it must be called before any of the other methods.

func (*Client) Mail

```
func (c *Client) Mail(from string) error
```

Mail issues a MAIL command to the server using the provided email address. If the server supports the 8BITMIME extension, Mail adds the BODY=8BITMIME parameter. This initiates a mail transaction and is followed by one or more Rcpt calls.

func (*Client) Noop

```
func (c *Client) Noop() error
```

Noop sends the NOOP command to the server. It does nothing but check that the connection to the server is okay.

func (*Client) Quit

```
func (c *Client) Quit() error
```

Quit sends the QUIT command and closes the connection to the server.

func (*Client) Rcpt

```
func (c *Client) Rcpt(to string) error
```

Rcpt issues a RCPT command to the server using the provided email address. A call to Rcpt must be preceded by a call to Mail and may be followed by a Data call or another Rcpt call.

func (*Client) Reset

```
func (c *Client) Reset() error
```

Reset sends the RSET command to the server, aborting the current mail transaction.

func (*Client) StartTLS

```
func (c *Client) StartTLS(config *tls.Config) error
```

StartTLS sends the STARTTLS command and encrypts all further communication. Only servers that advertise the STARTTLS extension support this function.

func (*Client) TLSConnectionState

```
func (c *Client) TLSConnectionState() (state tls.ConnectionState, ok bool)
```

TLSConnectionState returns the client's TLS connection state. The return values are their zero values if StartTLS did not succeed.

func (*Client) Verify

```
func (c *Client) Verify(addr string) error
```

Verify checks the validity of an email address on the server. If Verify returns nil, the address is valid. A non-nil return does not necessarily indicate an invalid address. Many servers will not verify addresses for security reasons.

type ServerInfo

```
type ServerInfo struct {
    Name string   // SMTP server name
    TLS  bool     // using TLS, with valid certificate for Name
    Auth []string // advertised authentication mechanisms
}
```

ServerInfo records information about an SMTP server.