NAME

Net::POP3 - Post Office Protocol 3 Client class (RFC1939)

SYNOPSIS

```
use Net::POP3;

# Constructors
$pop = Net::POP3->new('pop3host');
$pop = Net::POP3->new('pop3host', Timeout => 60);
$pop = Net::POP3->new('pop3host', SSL => 1, Timeout => 60);

if ($pop->login($username, $password) > 0) {
  my $msgnums = $pop->list; # hashref of msgnum => size
  foreach my $msgnum (keys %$msgnums) {
    my $msg = $pop->get($msgnum);
    print @$msg;
    $pop->delete($msgnum);
  }
}

$pop->quit;
```

DESCRIPTION

This module implements a client interface to the POP3 protocol, enabling a perl5 application to talk to POP3 servers. This documentation assumes that you are familiar with the POP3 protocol described in RFC1939.

A new Net::POP3 object must be created with the *new* method. Once this has been done, all POP3 commands are accessed via method calls on the object.

The Net::POP3 class is a subclass of Net::Cmd and IO::Socket::INET.

CONSTRUCTOR

new ([HOST] [, OPTIONS])

This is the constructor for a new Net::POP3 object. `HOST` is the name of the remote host to which an POP3 connection is required.

`HOST` is optional. If `HOST` is not given then it may instead be passed as the `Host` option described below. If neither is given then the `POP3_Hosts` specified in `Net::Config` will be used.

`OPTIONS` are passed in a hash like fashion, using key and value pairs. Possible options are:

Host - POP3 host to connect to. It may be a single scalar, as defined for the `PeerAddr` option in *IO::Socket::INET*, or a reference to an array with hosts to try in turn. The *host* method will return the value which was used to connect to the host.

Port - port to connect to. Default - 110 for plain POP3 and 995 for POP3s (direct SSL).

SSL - If the connection should be done from start with SSL, contrary to later upgrade with `starttls`. You can use SSL arguments as documented in *IO::Socket::SSL*, but it will usually use the right arguments already.

ResvPort - If given then the socket for the `Net::POP3` object will be bound to the local port given using `bind` when the socket is created.

Timeout - Maximum time, in seconds, to wait for a response from the POP3 server (default: 120)

Debug - Enable debugging information

METHODS

Unless otherwise stated all methods return either a *true* or *false* value, with *true* meaning that the operation was a success. When a method states that it returns a value, failure will be returned as *undef* or an empty list.

`Net::POP3` inherits from `Net::Cmd` so methods defined in `Net::Cmd` may be used to send commands to the remote POP3 server in addition to the methods documented here.

host ()

Returns the value used by the constructor, and passed to IO::Socket::INET, to connect to the host.

auth (USERNAME, PASSWORD)

Attempt SASL authentication.

user (USER)

Send the USER command.

pass (PASS)

Send the PASS command. Returns the number of messages in the mailbox.

login ([USER [, PASS]])

Send both the USER and PASS commands. If `PASS` is not given the `Net::POP3` uses `Net::Netrc` to lookup the password using the host and username. If the username is not specified then the current user name will be used.

Returns the number of messages in the mailbox. However if there are no messages on the server the string `"0E0"` will be returned. This is will give a true value in a boolean context, but zero in a numeric context.

If there was an error authenticating the user then *undef* will be returned.

starttls (SSLARGS)

Upgrade existing plain connection to SSL. You can use SSL arguments as documented in *IO::Socket::SSL*, but it will usually use the right arguments already.

apop ([USER [, PASS]])

Authenticate with the server identifying as `USER` with password `PASS`. Similar to *login*, but the password is not sent in clear text.

To use this method you must have the Digest::MD5 or the MD5 module installed, otherwise this method will return *undef*.

banner ()

Return the sever's connection banner

capa ()

Return a reference to a hash of the capabilities of the server. APOP is added as a pseudo capability. Note that I've been unable to find a list of the standard capability values, and some appear to be multi-word and some are not. We make an attempt at intelligently parsing them, but it may not be correct.

capabilities ()

Just like capa, but only uses a cache from the last time we asked the server, so as to avoid asking more than once.

top (MSGNUM [, NUMLINES])

Get the header and the first `NUMLINES` of the body for the message `MSGNUM`. Returns a

reference to an array which contains the lines of text read from the server.

list ([MSGNUM])

If called with an argument the `list` returns the size of the message in octets.

If called without arguments a reference to a hash is returned. The keys will be the `MSGNUM`'s of all undeleted messages and the values will be their size in octets.

get (MSGNUM [, FH])

Get the message `MSGNUM` from the remote mailbox. If `FH` is not given then get returns a reference to an array which contains the lines of text read from the server. If `FH` is given then the lines returned from the server are printed to the filehandle `FH`.

getfh (MSGNUM)

As per get(), but returns a tied filehandle. Reading from this filehandle returns the requested message. The filehandle will return EOF at the end of the message and should not be reused.

last ()

Returns the highest `MSGNUM` of all the messages accessed.

popstat ()

Returns a list of two elements. These are the number of undeleted elements and the size of the mbox in octets.

ping (USER)

Returns a list of two elements. These are the number of new messages and the total number of messages for `USER`.

uidl ([MSGNUM])

Returns a unique identifier for `MSGNUM` if given. If `MSGNUM` is not given `uidl` returns a reference to a hash where the keys are the message numbers and the values are the unique identifiers.

delete (MSGNUM)

Mark message `MSGNUM` to be deleted from the remote mailbox. All messages that are marked to be deleted will be removed from the remote mailbox when the server connection closed.

reset ()

Reset the status of the remote POP3 server. This includes resetting the status of all messages to not be deleted.

quit ()

Quit and close the connection to the remote POP3 server. Any messages marked as deleted will be deleted from the remote mailbox.

can_inet6 ()

Returns whether we can use IPv6.

can_ssl ()

Returns whether we can use SSL.

NOTES

If a `Net::POP3` object goes out of scope before `quit` method is called then the `reset` method will called before the connection is closed. This means that any messages marked to be deleted will not be.

SEE ALSO

Net::Netrc, *Net::Cmd*, *IO::Socket::SSL*

AUTHOR

Graham Barr <*gbarr@pobox.com*>

Steve Hay <*shay@cpan.org*> is now maintaining libnet as of version 1.22_02

COPYRIGHT