21.9. `poplib` — POP3 protocol client¶

This module defines a class, `POP3`, which encapsulates a connection to a POP3 server and implements the protocol as defined in **RFC 1725**. The `POP3` class supports both the minimal and optional command sets. Additionally, this module provides a class `POP3_SSL`, which provides support for connecting to POP3 servers that use SSL as an underlying protocol layer.

Note that POP3, though widely supported, is obsolescent. The implementation quality of POP3 servers varies widely, and too many are quite poor. If your mailserver supports IMAP, you would be better off using the `imaplib.IMAP4` class, as IMAP servers tend to be better implemented.

A single class is provided by the `poplib` module:

class `poplib.POP3`(*host*[, *port*[, *timeout*]])¶

This class implements the actual POP3 protocol. The connection is created when the instance is initialized. If *port* is omitted, the standard POP3 port (110) is used. The optional *timeout* parameter specifies a timeout in seconds for the connection attempt (if not specified, the global default timeout setting will be used).

Changed in version 2.6: *timeout* was added.

class `poplib.POP3_SSL`(*host*[, *port*[, *keyfile*[, *certfile*]]])¶

This is a subclass of `POP3` that connects to the server over an SSL encrypted socket. If *port* is not specified, 995, the standard POP3-over-SSL port is used. *keyfile* and *certfile* are also optional - they can contain a PEM formatted private key and certificate chain file for the SSL connection.

New in version 2.4.

One exception is defined as an attribute of the `poplib` module:

exception `poplib.error_proto`¶

Exception raised on any errors from this module (errors from `socket` module are not caught). The reason for the exception is passed to the constructor as a string.

See also

Module `imaplib`

The standard Python IMAP module.

Frequently Asked Questions About Fetchmail

The FAQ for the **fetchmail** POP/IMAP client collects information on POP3 server variations and RFC noncompliance that may be useful if you need to write an application based on the POP protocol.

21.9.1. POP3 Objects¶

All POP3 commands are represented by methods of the same name, in lower-case; most return the response text sent by the server.

An `POP3` instance has the following methods:

`POP3.set_debuglevel`(*level*)¶

Set the instance's debugging level. This controls the amount of debugging output printed. The default, `0`, produces no debugging output. A value of `1` produces a moderate amount of debugging output, generally a single line per request. A value of `2` or higher produces the maximum amount of debugging output, logging each line sent and received on the control connection.

`POP3.getwelcome`()¶

Returns the greeting string sent by the POP3 server.

`POP3.user`(*username*)¶

Send user command, response should indicate that a password is required.

`POP3.pass_`(*password*)¶

Send password, response includes message count and mailbox size. Note: the mailbox on the server is locked until `quit()` is called.

`POP3.apop`(*user*, *secret*)¶

Use the more secure APOP authentication to log into the POP3 server.

`POP3.rpop`(*user*)¶
Use RPOP authentication (similar to UNIX r-commands) to log into POP3 server.

`POP3.stat`()¶
Get mailbox status. The result is a tuple of 2 integers: `(message count, mailbox size)`.

`POP3.list`([*which*])¶
Request message list, result is in the form `(response, ['mesg_num octets', ...], octets)`. If *which* is set, it is the message to list.

`POP3.retr`(*which*)¶
Retrieve whole message number *which*, and set its seen flag. Result is in form `(response, ['line', ...], octets)`.

`POP3.dele`(*which*)¶
Flag message number *which* for deletion. On most servers deletions are not actually performed until QUIT (the major exception is Eudora QPOP, which deliberately violates the RFCs by doing pending deletes on any disconnect).

`POP3.rset`()¶
Remove any deletion marks for the mailbox.

`POP3.noop`()¶
Do nothing. Might be used as a keep-alive.

`POP3.quit`()¶
Signoff: commit changes, unlock mailbox, drop connection.

`POP3.top`(*which*, *howmuch*)¶

Retrieves the message header plus *howmuch* lines of the message after the header of message number *which*. Result is in form `(response, ['line', ...], octets)`.

The POP3 TOP command this method uses, unlike the RETR command, doesn't set the message's seen flag; unfortunately, TOP is poorly specified in the RFCs and is frequently broken in off-brand servers. Test this method by hand against the POP3 servers you will use before trusting it.

`POP3.uidl`([*which*])¶
Return message digest (unique id) list. If *which* is specified, result contains the unique id for that message in the form `'response mesgnum uid`, otherwise result is list `(response, ['mesgnum uid', ...], octets)`.

Instances of `POP3_SSL` have no additional methods. The interface of this subclass is identical to its parent.

21.9.2. POP3 Example¶

Here is a minimal example (without error checking) that opens a mailbox and retrieves and prints all messages:

```
import getpass, poplib

M = poplib.POP3('localhost')
M.user(getpass.getuser())
M.pass_(getpass.getpass())
numMessages = len(M.list()[1])
for i in range(numMessages):
   for j in M.retr(i+1)[1]:
       print j
```

At the end of the module, there is a test section that contains a more extensive example of usage.