Network Working Group
Request for Comments: 1081

M. Rose
TWG

November 1988

Post Office Protocol - Version 3

Status of this Memo

This memo suggests a simple method for workstations to dynamically access mail from a mailbox server. This RFC specifies a proposed protocol for the Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

This memo is based on RFC 918 (since revised as RFC 937). Although similar in form to the original Post Office Protocol (POP) proposed for the Internet community, the protocol discussed in this memo is similar in spirit to the ideas investigated by the MZnet project at the University of California, Irvine.

Further, substantial work was done on examining POP in a PC-based environment. This work, which resulted in additional functionality in this protocol, was performed by the ACIS Networking Systems Group at Stanford University. The author gratefully acknowledges their interest.

Introduction

On certain types of smaller nodes in the Internet it is often impractical to maintain a message transport system (MTS). For example, a workstation may not have sufficient resources (cycles, disk space) in order to permit a SMTP server and associated local mail delivery system to be kept resident and continuously running. Similarly, it may be expensive (or impossible) to keep a personal computer interconnected to an IP-style network for long amounts of time (the node is lacking the resource known as "connectivity").

Despite this, it is often very useful to be able to manage mail on these smaller nodes, and they often support a user agent (UA) to aid the tasks of mail handling. To solve this problem, a node which can support an MTS entity offers a maildrop service to these less endowed nodes. The Post Office Protocol - Version 3 (POP3) is intended to permit a workstation to dynamically access a maildrop on a server host in a useful fashion. Usually, this means that the POP3 is used to allow a workstation to retrieve mail that the server is holding for it.

For the remainder of this memo, the term "client host" refers to a host making use of the POP3 service, while the term "server host" refers to a host which offers the POP3 service.

A Short Digression

This memo does not specify how a client host enters mail into the transport system, although a method consistent with the philosophy of this memo is presented here:

> When the user agent on a client host wishes to enter a message into the transport system, it establishes an SMTP connection to its relay host (this relay host could be, but need not be, the POP3 server host for the client host).

If this method is followed, then the client host appears to the MTS as a user agent, and should NOT be regarded as a "trusted" MTS entity in any sense whatsoever. This concept, along with the role of the POP3 as a part of a split-UA model is discussed later in this memo.

Initially, the server host starts the POP3 service by listening on TCP port 110. When a client host wishes to make use of the service, it establishes a TCP connection with the server host. When the connection is established, the POP3 server sends a greeting. The client and POP3 server then exchange commands and responses (respectively) until the connection is closed or aborted.

Commands in the POP3 consist of a keyword possibly followed by an argument. All commands are terminated by a CRLF pair.

Responses in the POP3 consist of a success indicator and a keyword possibly followed by additional information. All responses are terminated by a CRLF pair. There are currently two success indicators: positive ("+OK") and negative ("-ERR").

Responses to certain commands are multi-line. In these cases, which are clearly indicated below, after sending the first line of the response and a CRLF, any additional lines are sent, each terminated by a CRLF pair. When all lines of the response have been sent, a final line is sent, consisting of a termination octet (decimal code 046, ".") and a CRLF pair. If any line of the multi-line response begins with the termination octet, the line is "byte-stuffed" by pre-pending the termination octet to that line of the response. Hence a multi-line response is terminated with the five octets "CRLF.CRLF". When examining a multi-line response, the client checks to see if the line begins with the termination octet. If so and if octets other than CRLF follow, the the first octet of the line (the termination octet) is stripped away. If so and if CRLF immediately

follows the termination character, then the response from the POP server is ended and the line containing ".CRLF" is not considered part of the multi-line response.

A POP3 session progresses through a number of states during its lifetime. Once the TCP connection has been opened and the POP3 server has sent the greeting, the session enters the AUTHORIZATION state. In this state, the client must identify itself to the POP3 server. Once the client has successfully done this, the server acquires resources associated with the client's maildrop, and the session enters the TRANSACTION state. In this state, the client requests actions on the part of the POP3 server. When the client has finished its transactions, the session enters the UPDATE state. In this state, the POP3 server releases any resources acquired during the TRANSACTION state and says goodbye. The TCP connection is then closed.

The AUTHORIZATION State

Once the TCP connection has been opened by a POP3 client, the POP3 server issues a one line greeting. This can be any string terminated by CRLF. An example might be:

S. +OK dewey POP3 server ready (Comments to: PostMaster@UDEL.EDU)

Note that this greeting is a POP3 reply. The POP3 server should always give a positive response as the greeting.

The POP3 session is now in the AUTHORIZATION state. The client must now issue the USER command. If the POP3 server responds with a positive success indicator ("+OK"), then the client may issue either the PASS command to complete the authorization, or the QUIT command to terminate the POP3 session. If the POP3 server responds with a negative success indicator ("-ERR") to the USER command, then the client may either issue a new USER command or may issue the QUIT command.

When the client issues the PASS command, the POP3 server uses the argument pair from the USER and PASS commands to determine if the client should be given access to the appropriate maildrop. If so, the POP3 server then acquires an exclusive-access lock on the maildrop. If the lock is successfully acquired, the POP3 server parses the maildrop into individual messages (read note below), determines the last message (if any) present in the maildrop that was referenced by the RETR command, and responds with a positive success indicator. The POP3 session now enters the TRANSACTION state. If the lock can not be acquired or the client should is denied access to the appropriate maildrop or the maildrop can't be parsed for some

reason, the POP3 server responds with a negative success indicator. (If a lock was acquired but the POP3 server intends to respond with a negative success indicator, the POP3 server must release the lock prior to rejecting the command.) At this point, the client may either issue a new USER command and start again, or the client may issue the QUIT command.

> NOTE: Minimal implementations of the POP3 need only be able to break a maildrop into its component messages; they need NOT be able to parse individual messages. More advanced implementations may wish to have this capability, for reasons discussed later.

After the POP3 server has parsed the maildrop into individual messages, it assigns a message-id to each message, and notes the size of the message in octets. The first message in the maildrop is assigned a message-id of "1", the second is assigned "2", and so on, so that the n'th message in a maildrop is assigned a message-id of "n". In POP3 commands and responses, all message-id's and message sizes are expressed in base-10 (i.e., decimal).

It sets the "highest number accessed" to be that of the last message referenced by the RETR command.

Here are summaries for the three POP3 commands discussed thus far:

```
        USER name
            Arguments: a server specific user-id (required)
            Restrictions: may only be given in the AUTHORIZATION
                state after the POP3 greeting or after an
                unsuccessful USER or PASS command
            Possible Responses:
                +OK name is welcome here
                -ERR never heard of name
            Examples:
                C:    USER mrose
                S:    +OK mrose is a real hoopy frood
                  ...
                C:    USER frated
                S:    -ERR sorry, frated doesn't get his mail here

        PASS string
            Arguments: a server/user-id specific password (required)
            Restrictions: may only be given in the AUTHORIZATION
                state after a successful USER command
            Possible Responses:
                +OK maildrop locked and ready
                -ERR invalid password
```

```
            -ERR unable to lock maildrop
        Examples:
            C:    USER mrose
            S:    +OK mrose is a real hoopy frood
            C:    PASS secret
            S:    +OK mrose's maildrop has 2 messages
                  (320 octets)
              ...
            C:    USER mrose
            S:    +OK mrose is a real hoopy frood
            C:    PASS secret
            S:    -ERR unable to lock mrose's maildrop, file
                  already locked

    QUIT
        Arguments: none
        Restrictions: none
        Possible Responses:
            +OK
        Examples:
            C:    QUIT
            S:    +OK dewey POP3 server signing off
```

The TRANSACTION State

Once the client has successfully identified itself to the POP3 server and the POP3 server has locked and burst the appropriate maildrop, the POP3 session is now in the TRANSACTION state. The client may now issue any of the following POP3 commands repeatedly. After each command, the POP3 server issues a response. Eventually, the client issues the QUIT command and the POP3 session enters the UPDATE state.

Here are the POP3 commands valid in the TRANSACTION state:

```
    STAT
        Arguments: none
        Restrictions: may only be given in the TRANSACTION state.
        Discussion:

          The POP3 server issues a positive response with a line
          containing information for the maildrop.  This line is
          called a "drop listing" for that maildrop.

          In order to simplify parsing, all POP3 servers are
          required to use a certain format for drop listings.
          The first octets present must indicate the number of
          messages in the maildrop.  Following this is the size
```

of the maildrop in octets. This memo makes no requirement on what follows the maildrop size. Minimal implementations should just end that line of the response with a CRLF pair. More advanced implementations may include other information.

NOTE: This memo STRONGLY discourages implementations from supplying additional information in the drop listing. Other, optional, facilities are discussed later on which permit the client to parse the messages in the maildrop.

Note that messages marked as deleted are not counted in either total.

Possible Responses:
+OK nn mm
Examples:
C: STAT
S: +OK 2 320

LIST [msg]
Arguments: a message-id (optionally) If a message-id is given, it may NOT refer to a message marked as deleted.
Restrictions: may only be given in the TRANSACTION state.
Discussion:

If an argument was given and the POP3 server issues a positive response with a line containing information for that message. This line is called a "scan listing" for that message.

If no argument was given and the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, for each message in the maildrop, the POP3 server responds with a line containing information for that message. This line is called a "scan listing" for that message.

In order to simplify parsing, all POP3 servers are required to use a certain format for scan listings. The first octets present must be the message-id of the message. Following the message-id is the size of the message in octets. This memo makes no requirement on what follows the message size in the scan listing. Minimal implementations should just end that line of

the response with a CRLF pair. More advanced implementations may include other information, as parsed from the message.

NOTE: This memo STRONGLY discourages implementations from supplying additional information in the scan listing. Other, optional, facilities are discussed later on which permit the client to parse the messages in the maildrop.

Note that messages marked as deleted are not listed.

Possible Responses:
+OK scan listing follows
-ERR no such message
Examples:

```
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
  ...
C:    LIST 2
S:    +OK 2 200
  ...
C:    LIST 3
S:    -ERR no such message, only 2 messages in
      maildrop
```

RETR msg
Arguments: a message-id (required) This message-id may NOT refer to a message marked as deleted.
Restrictions: may only be given in the TRANSACTION state.
Discussion:

If the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, the POP3 server sends the message corresponding to the given message-id, being careful to byte-stuff the termination character (as with all multi-line responses).

If the number associated with this message is higher than the "highest number accessed" in the maildrop, the POP3 server updates the "highest number accessed" to the number associated with this message.

```
        Possible Responses:
            +OK message follows
            -ERR no such message
        Examples:
            C:    RETR 1
            S:    +OK 120 octets
            S:    <the POP3 server sends the entire message here>
            S:    .

    DELE msg
        Arguments: a message-id (required)  This message-id
            may NOT refer to a message marked as deleted.
        Restrictions: may only be given in the TRANSACTION state.
        Discussion:

          The POP3 server marks the message as deleted.  Any
          future reference to the message-id associated with the
          message in a POP3 command generates an error.  The POP3
          server does not actually delete the message until the
          POP3 session enters the UPDATE state.

          If the number associated with this message is higher
          than the "highest number accessed" in the maildrop,
          the POP3 server updates the "highest number accessed"
          to the number associated with this message.

        Possible Responses:
            +OK message deleted
            -ERR no such message
        Examples:
            C:    DELE 1
            S:    +OK message 1 deleted
              ...
            C:    DELE 2
            S:    -ERR message 2 already deleted

    NOOP
        Arguments: none
        Restrictions: may only be given in the TRANSACTION state.
        Discussion:

          The POP3 server does nothing, it merely replies with a
          positive response.

        Possible Responses:
            +OK
```

Examples:

```
C:    NOOP
S:    +OK
```

LAST

Arguments: none

Restrictions: may only be issued in the TRANSACTION state.

Discussion:

The POP3 server issues a positive response with a line containing the highest message number which accessed. Zero is returned in case no message in the maildrop has been accessed during previous transactions. A client may thereafter infer that messages, if any, numbered greater than the response to the LAST command are messages not yet accessed by the client.

Possible Response:

```
+OK nn
```

Examples:

```
C:      STAT
S:      +OK 4 320
C:      LAST
S:      +OK 1
C:      RETR 3
S:      +OK 120 octets
S:      <the POP3 server sends the entire message
        here>
S:      .
C:      LAST
S:      +OK 3
C:      DELE 2
S:      +OK message 2 deleted
C:      LAST
S:      +OK 3
C:      RSET
S:      +OK
C:      LAST
S:      +OK 1
```

RSET

Arguments: none

Restrictions: may only be given in the TRANSACTION state.

Discussion:

If any messages have been marked as deleted by the POP3

server, they are unmarked. The POP3 server then replies with a positive response. In addition, the "highest number accessed" is also reset to the value determined at the beginning of the POP3 session.

Possible Responses:
+OK
Examples:
C: RSET
S: +OK maildrop has 2 messages (320 octets)

The UPDATE State

When the client issues the QUIT command from the TRANSACTION state, the POP3 session enters the UPDATE state. (Note that if the client issues the QUIT command from the AUTHORIZATION state, the POP3 session terminates but does NOT enter the UPDATE state.)

QUIT
Arguments: none
Restrictions: none
Discussion:

The POP3 server removes all messages marked as deleted from the maildrop. It then releases the exclusive-access lock on the maildrop and replies as to the success of these operations. The TCP connection is then closed.

Possible Responses:
+OK
Examples:
C: QUIT
S: +OK dewey POP3 server signing off (maildrop empty)
...
C: QUIT
S: +OK dewey POP3 server signing off (2 messages left)
...

Optional POP3 Commands

The POP3 commands discussed above must be supported by all minimal implementations of POP3 servers.

The optional POP3 commands described below permit a POP3 client greater freedom in message handling, while preserving a simple POP3 server implementation.

NOTE: This memo STRONGLY encourages implementations to support these commands in lieu of developing augmented drop and scan listings. In short, the philosophy of this memo is to put intelligence in the part of the POP3 client and not the POP3 server.

TOP msg n

Arguments: a message-id (required) and a number. This message-id may NOT refer to a message marked as deleted.

Restrictions: may only be given in the TRANSACTION state.

Discussion:

If the POP3 server issues a positive response, then the response given is multi-line. After the initial +OK, the POP3 server sends the headers of the message, the blank line separating the headers from the body, and then the number of lines indicated message's body, being careful to byte-stuff the termination character (as with all multi-line responses).

Note that if the number of lines requested by the POP3 client is greater than than the number of lines in the body, then the POP3 server sends the entire message.

Possible Responses:

+OK top of message follows

-ERR no such message

Examples:

```
C:    TOP 10
S:    +OK
S:    <the POP3 server sends the headers of the
       message, a blank line, and the first 10 lines
       of the body of the message>
S:    .
  ...
C:    TOP 100
S:    -ERR no such message
```

RPOP user

Arguments: a client specific user-id (required)

Restrictions: may only be given in the AUTHORIZATION state after a successful USER command; in addition, may only be given if the client used a reserved

(privileged) TCP port to connect to the server.

Discussion:

The RPOP command may be used instead of the PASS command to authenticate access to the maildrop. In order for this command to be successful, the POP3 client must use a reserved TCP port (port < 1024) to connect tothe server. The POP3 server uses the argument pair from the USER and RPOP commands to determine if the client should be given access to the appropriate maildrop. Unlike the PASS command however, the POP3 server considers if the remote user specified by the RPOP command who resides on the POP3 client host is allowed to access the maildrop for the user specified by the USER command (e.g., on Berkeley UNIX, the .rhosts mechanism is used). With the exception of this differing in authentication, this command is identical to the PASS command.

Note that the use of this feature has allowed much wider penetration into numerous hosts on local networks (and sometimes remote networks) by those who gain illegal access to computers by guessing passwords or otherwise breaking into the system.

Possible Responses:

+OK maildrop locked and ready
-ERR permission denied

Examples:

```
C:    USER mrose
S:    +OK mrose is a real hoopy frood
C:    RPOP mrose
S:    +OK mrose's maildrop has 2 messages (320
      octets)
```

Minimal POP3 Commands:

```
USER name               valid in the AUTHORIZATION state
PASS string
QUIT

STAT                    valid in the TRANSACTION state
LIST [msg]
RETR msg
DELE msg
NOOP
LAST
RSET
```

```
        QUIT                    valid in the UPDATE state

    Optional POP3 Commands:
        RPOP user               valid in the AUTHORIZATION state

        TOP msg n               valid in the TRANSACTION state

    POP3 Replies:
        +OK
        -ERR
```

Note that with the exception of the STAT command, the reply given by the POP3 server to any command is significant only to "+OK" and "-ERR". Any text occurring after this reply may be ignored by the client.

Example POP3 Session

```
S: <wait for connection on TCP port 110>
    ...
C: <open connection>
S:    +OK dewey POP3 server ready (Comments to: PostMaster@UDEL.EDU)
C:    USER mrose
S:    +OK mrose is a real hoopy frood
C:    PASS secret
S:    +OK mrose's maildrop has 2 messages (320 octets)
C:    STAT
S:    +OK 2 320
C:    LIST
S:    +OK 2 messages (320 octets)
S:    1 120
S:    2 200
S:    .
C:    RETR 1
S:    +OK 120 octets
S:    <the POP3 server sends message 1>
S:    .
C:    DELE 1
S:    +OK message 1 deleted
C:    RETR 2
S:    +OK 200 octets
S:    <the POP3 server sends message 2>
S:    .
C:    DELE 2
S:    +OK message 2 deleted
C:    QUIT
```

```
S:    +OK dewey POP3 server signing off (maildrop empty)
C:  <close connection>
S:  <wait for next connection>
```

Message Format

All messages transmitted during a POP3 session are assumed to conform to the standard for the format of Internet text messages [RFC822].

It is important to note that the byte count for a message on the server host may differ from the octet count assigned to that message due to local conventions for designating end-of-line. Usually, during the AUTHORIZATION state of the POP3 session, the POP3 client can calculate the size of each message in octets when it parses the maildrop into messages. For example, if the POP3 server host internally represents end-of-line as a single character, then the POP3 server simply counts each occurrence of this character in a message as two octets. Note that lines in the message which start with the termination octet need not be counted twice, since the POP3 client will remove all byte-stuffed termination characters when it receives a multi-line response.

The POP and the Split-UA model

The underlying paradigm in which the POP3 functions is that of a split-UA model. The POP3 client host, being a remote PC based workstation, acts solely as a client to the message transport system. It does not provide delivery/authentication services to others. Hence, it is acting as a UA, on behalf of the person using the workstation. Furthermore, the workstation uses SMTP to enter mail into the MTS.

In this sense, we have two UA functions which interface to the message transport system: Posting (SMTP) and Retrieval (POP3). The entity which supports this type of environment is called a split-UA (since the user agent is split between two hosts which must interoperate to provide these functions).

> ASIDE: Others might term this a remote-UA instead. There are arguments supporting the use of both terms.

This memo has explicitly referenced TCP as the underlying transport agent for the POP3. This need not be the case. In the MZnet split-UA, for example, personal micro-computer systems are used which do not have IP-style networking capability. To connect to the POP3 server host, a PC establishes a terminal connection using some simple protocol (PhoneNet). A program on the PC drives the connection, first establishing a login session as a normal user. The login shell

for this pseudo-user is a program which drives the other half of the terminal protocol and communicates with one of two servers. Although MZnet can support several PCs, a single pseudo-user login is present on the server host. The user-id and password for this pseudo-user login is known to all members of MZnet. Hence, the first action of the login shell, after starting the terminal protocol, is to demand a USER/PASS authorization pair from the PC. This second level of authorization is used to ascertain who is interacting with the MTS. Although the server host is deemed to support a "trusted" MTS entity, PCs in MZnet are not. Naturally, the USER/PASS authorization pair for a PC is known only to the owner of the PC (in theory, at least).

After successfully verifying the identity of the client, a modified SMTP server is started, and the PC posts mail with the server host. After the QUIT command is given to the SMTP server and it terminates, a modified POP3 server is started, and the PC retrieves mail from the server host. After the QUIT command is given to the POP3 server and it terminates, the login shell for the pseudo-user terminates the terminal protocol and logs the job out. The PC then closes the terminal connection to the server host.

The SMTP server used by MZnet is modified in the sense that it knows that it’s talking to a user agent and not a "trusted" entity in the message transport system. Hence, it does performs the validation activities normally performed by an entity in the MTS when it accepts a message from a UA.

The POP3 server used by MZnet is modified in the sense that it does not require a USER/PASS combination before entering the TRANSACTION state. The reason for this (of course) is that the PC has already identified itself during the second-level authorization step described above.

> NOTE: Truth in advertising laws require that the author of this memo state that MZnet has not actually been fully implemented. The concepts presented and proven by the project led to the notion of the MZnet split-slot model. This notion has inspired the split-UA concept described in this memo, led to the author’s interest in the POP, and heavily influenced the the description of the POP3 herein.

In fact, some UAs present in the Internet already support the notion of posting directly to an SMTP server and retrieving mail directly from a POP server, even if the POP server and client resided on the same host!

> ASIDE: this discussion raises an issue which this memo

purposedly avoids: how does SMTP know that it's talking to a "trusted" MTS entity?

References

[MZnet] Stefferud, E., J. Sweet, and T. Domae, "MZnet: Mail Service for Personal Micro-Computer Systems", Proceedings, IFIP 6.5 International Conference on Computer Message Systems, Nottingham, U.K., May 1984.

[RFC821] Postel, J., "Simple Mail Transfer Protocol", USC/Information Sciences Institute, August 1982.

[RFC822] Crocker, D., "Standard for the Format of ARPA-Internet Text Messages", University of Delaware, August 1982.

[RFC937] Butler, M., J. Postel, D. Chase, J. Goldberger, and J. Reynolds, "Post Office Protocol - Version 2", RFC 937, USC/Information Sciences Institute, February 1985.

[RFC1010] Reynolds, J., and J. Postel, "Assigned Numbers", RFC 1010, USC/Information Sciences Institute, May 1987.

Author's Address:

Marshall Rose
The Wollongong Group
1129 San Antonio Rd.
Palo Alto, California 94303

Phone: (415) 962-7100

Email: MRose@TWG.COM