POP3 External Edition Protocol(PEEP)

1 Introduction

The traditional Post Office Protocol version 3 (POP3) protocol [RFC 1939] has served as a widely adopted standard for retrieving email messages from a remote server. However, it does not offer a way to edit the server configuration nor monitoring metrics in real time.

The POP3 External Edition Protocol (PEEP) is a text oriented TCP-based protocol designed to enhance the functionality of a POP3 server. PEEP enables the configuration and real-time monitoring of various metrics associated with a POP3 server, allowing administrators to efficiently manage and optimize their email systems.

1.1 Considerations

Along this document, whenever a command or response syntax is defined, its arguments will be represented by its key name between ‘<’, ‘>’. This means that the whole ‘<key>’ must be replaced with the argument value.

On the other hand, the meaning behind a response will be specified between parentheses, but this is not part of the response syntax definition.

Throughout this document, the meaning of modals such as “may” or “must” shall be interpreted as described in the RFC 2119 [RFC2119].

2 Basic Operation

Initially, the server host starts the PEEP service by listening on TCP port 2110. Whenever a client host attempts to make use of the service, it does so by establishing a TCP connection with the server host. The client may then send commands which the server will respond to. The transaction finishes when the connection is either closed or aborted.

There are only two states AUTHENTICATION and AUTHENTICATED. In the former only the ‘a’, ‘q’ and ‘cap?’ commands are available (these are described later in this document). All connections start in the AUTHENTICATION state.

The PEEP server can support multiple connections from different clients simultaneously.

Implementations may support command PIPELINING.

2.1 Command format

All commands in PEEP consist of 1 or more lowercase letters which may be followed by a special ascii character (‘+’, ‘-’, ‘=’ or ‘?’), from now on these will be referred to as the “commencement” of the command. Should the command use one or two arguments, then a single space character is placed right after the special character and between the arguments. All commands are ended with a CRLF (‘\r\n’).

The commencement of a command must not be longer than 8 characters, while each argument must not be longer than 128 characters. Therefore, (including space characters and CRLF) a command with two arguments will be at most 268 characters long.

2.2 Response Format

Responses can be single-line or multi-line, depending on the nature of the information to transmit. For both cases, each line must end with a CRLF and not be longer than 512 characters (including the CRLF).

Single-line responses must be either positive or negative. The former starts with a ‘+’ character, while the latter does so with a ‘-’ character. After this initial character, it may include the content of the answer, be it an integer, a string, or nothing at all. Finally, it ends the line with a CRLF as mentioned earlier.

Examples of single-line responses:

```
S: +
S: +2612
S: +~/Maildir/
```

Multi-line responses must always start with “+n” being ‘n’ an unsigned integer representing the number of lines that make up the response (without counting this first line). For example:

```
S: +3
S: alice
S: bob
S: charles
```

Negative responses are always single-line and represent errors in the execution of a command. There are two default negative responses that can be issued for any client input, regardless of the command or state. Therefore, they will not be mentioned in any command definitions in the following sections, but the reader should keep in mind that these default negative responses exist for any command.

```
UNKNOWN COMMAND ERROR
```

Response syntax: -0

Issued when the commencement of a command does not match any valid (in the current state) command's commencement. This also includes the case where a command sent by the client is empty.

INVALID COMMAND SYNTAX ERROR
Response syntax: -1

Issued when the commencement of a command matches a valid (in the current state) command's commencement, but either of the following situations is present:

- The argument count is smaller than necessary. (Exceeding arguments may be ignored)
- The argument format is incorrect
- There are more than one space characters between the commencement and the first argument, or between arguments.

For all commands where a value is returned in the commands response, this value may be volatile, meaning that it resets whenever the server shuts down.

3 The AUTHENTICATION State

As soon as the TCP connection has been established, the PEEP session is in AUTHENTICATION state. In the said state the client can only identify itself by using the 'a' command with their username and password as described in the following section. The fact that all PEEP servers must support this authentication mechanism should go without saying. It is not obvious, however, that the PEEP protocol may be extended in the future by adding more authentication mechanisms such as a key-based authentication.

The only commands available in this state are:

AUTHENTICATE(a)

Syntax: a <username> <password>

Arguments format:
username: string
password: string

Possible Positive Responses:
+ (authentication successful)

Possible Negative Responses:

-2 (incorrect username or password)

Considerations:

The username and password should be known by an administrator of the server and may not be the same as the POP3 server users. Both username and password can not contain space characters in them.

This protocol does not go over the definition of this administrator's username and password, but it is recommended to do this externally to the protocol. One or more pairs of username and password may be defined, but all sessions will share the same functionality.

Examples:

C: a bob password_of_bob
S: +

C: a bob not_password_of_bob
S: -2

LIST CAPABILITIES(cap?)

Syntax: cap?

OK Response:

+<amount_of_commands>
<command_1>
<command_2>
<command_3>
…
<command_n>

Formats of Positive Response Value:

amount_of_commands: int
command_i: string

Error Response:

Only defaults

Considerations:

The response must include a list of the commencement of:

- all commands available for authentication (like 'a')
- all commands that require an integer as argument, indicating the valid number range with the format "<min>-<max>" after the command's commencement.

- all commands implemented that were defined in future extensions of this protocol.

Example:
C: cap?
S: +3
S: a
S: c=1-500
S: t=0-300

QUIT(q)

Syntax: q

Possible Positive Responses:
+ (exit successful)

Possible Negative Responses:
only default

Considerations:
This command must close the connection after answering a positive response.

Examples:
C: q
S: +

4 The AUTHENTICATED State

Once the PEEP server has validated the previously described authentication, the client is granted access to all commands. By now, the client will find itself in the AUTHENTICATED state.

In this state all commands are available for use, except for the AUTHENTICATE command, obviously.

Here are the PEEP commands valid in the AUTHENTICATED state:

ADD USER(u+)

Syntax: u+ <username> <password>

Arguments format:
username: string
password: string

Possible Positive Responses:
+ (successful creation)

Possible Negative Responses:
-3 (username already in use)

Considerations:
It will add a user to the POP3 server if its username is not already in use.
This protocol does not support space characters in both username and password, as this character is being used as a separator for command arguments.

Examples:
C: u+ linus torvalds
S: -3
C: u+ unix torvalds
S: +

DELETE USER(u-)

Syntax: u- <username>

Argument format:
username: string

Possible Positive Responses:
+ (successful deletion)

Possible Negative Responses:
-4 (username does not exist)

Considerations:
If the user to be deleted is currently logged in, the POP3 server may not close its connection, but the user will not be able to log in anymore after the current connection is terminated.

Examples:
C: u+ linus torvalds
S: -3
C: u+ unix torvalds
S: +

SHOW USERS(u?)

Syntax: u?

Possible Positive Response:
+<amount_of_users>
<username_1>
<username_2>

```
            …
            <username_n>

      Formats of Positive Response Values:
            amount_of_users: int
            username_i: string

      Possible Negative Responses:
            Only defaults

      Considerations:
            It returns a multi-line response including the
            usernames from all POP3 server users.

      Example:
            C: u?
            S: +2
               linus
               unix



SET MAX CONNECTIONS(c=)

      Syntax:
            c= <new_max_connections>

      Possible Positive Response:
            new_max_connections: int

      Positive Positive Response:
            +     (new max connection limit set)

      Error Responses:
            -5    (invalid max count)

      Considerations:
            Only for this command and the one defined below
            (‘c=’ and ‘c?’), the server may consider both POP3
            and PEEP clients as connections that are limited by
            this maximum.

            If the new upper limit of connections is set to a
            value smaller than the current connection count,
            the POP3 server may not close the exceeding
            connections, but must not accept new connections
            until the current count drops below the new limit.

            The invalid max count error must be returned when
            the argument is an integer, but is not included in
```

the desired range. The desired range must be specified when the command is listed in the ‘h?’ command, as mentioned earlier.

Example:
C: c= 15
S: +

SHOW MAX CONNECTIONS(c?)

Syntax:
c?

Possible Positive Response:
+<amount_of_max_conections>

Formats of Positive Response Value:
amount_of_max_conections: int

Error Response:
Only defaults

Considerations:
The response includes the current limit of concurrent connections to the server.

SHOW MAILDIR(m?)

Syntax:
m?

OK Response:
+<absolute_path_to_the_mail_dir>

Formats of Positive Response Value:
absolute_path_to_the_mail_dir: string

Error Responses:
Only defaults

Considerations:
Show the absolute path to the main mail directory, where all the users’ mail directories are.

SET MAILDIR(m=)

Syntax:
m= <absolute_path>

Argument format:

absolute_path: string

Possible Positive Response:
+ (new maildir set)

Error Responses:
-6 (invalid path)

Considerations:
This command sets the directory path from which the POP3 server retrieves its emails. The directory change may apply only for new users, while currently connected users will work with the old directory until they disconnect.

This protocol does not support path values with space characters, nor CRLF.

The invalid path error should be returned when the string given does not match a path format or it is not possible to use this path as a mail directory.

SHOW TIMEOUT(t?)

Syntax
t?

Possible Positive Response:
+<timeout_in_seconds>

Formats of Positive Response Value:
timeout_in_seconds: int

Error Response:
Only defaults

Considerations:
The timeout is the maximum amount of time between client’s interactions with the POP3 server, before the connection is closed.

SET TIMEOUT (t=)

Syntax:
t= <new_time>

Argument Format:
new_time: int

Possible Positive Response:

```
            +     (timeout correctly changed)

      Error Response:
            -7    (invalid timeout)

      Considerations:
            The invalid timeout error should be returned when
            an undesired value is given for the new time. Each
            implementation may set different limits for the
            valid timeout range, but these limits must be
            specified when the command is listed in the 'cap?'
            command.

            It is recommended to use the value zero, to
            represent no timeout.

SHOW RETRIEVED BYTES(rb?)

      Syntax:
            rb?

      OK Response:
            +<retrieved_bytes_count>

      Formats of Positive Response Value:
            retrieved_bytes_count: int

      Error Response:
            Only defaults

      Considerations:
            The retrieved bytes count is defined as the sum of
            all the bytes sent to clients as answers to any
            POP3 command.

SHOW RETRIEVED EMAIL COUNT(re?)

      Syntax:
            re?

      OK Response:
            +<retrieved_emails_count>

      Formats of Positive Response Value:
            retrieved_emails_count: int

      Error Response:
            Only defaults

      Considerations:
```

This command counts the successful RETR commands made by all users of the POP3 server.

SHOW REMOVED EMAILS COUNT(xe?)

Syntax:
xe?

OK Response:
+<removed_emails_count>

Formats of Positive Response Value:
removed_emails_count: int

Error Response:
Only defaults

Considerations:
This command returns the amount of emails that have been effectively deleted in the POP3 UPDATE state, after being flagged for deletion with the DELE command.

SHOW CURRENT CONNECTIONS COUNT (cc?)

Syntax:
cc?

OK Response:
+<current_connections_count>

Formats of Positive Response Value:
current_connections_count: int

Error Response:
Only defaults

Considerations:
This command returns the amount of current POP3 client connections with the server, even if they have not been authorized.

SHOW CURRENT LOGGED IN USERS(cu?)

Syntax
cu?

OK Response:
+<amount_of_logged_in_users>

<username_1>
<username_2>
…
<username_n>

Formats of Positive Response Values:
amount_of_logged_in_users: int
username_i: string

Error Response:
Only default

Considerations:
This command returns a list of the usernames of all currently connected users that have been authorized in the POP3 server.

Example:
C: cu?
S: +3
linus
fulano
unix

SHOW HISTORICAL CONNECTIONS COUNT(hc?)

Syntax:
hc?

OK Response:
+<historical_connections>

Formats of Positive Response Value:
historical_connections: int

Error Response:
Only defaults

Example:
C: hc?
S: +23

Considerations:
This command only considers the amount of successful POP3 connections that were made to the server, even if they have not been authorized.

SHOW HISTORICAL LOGGED IN USERS COUNT (hu?)

Syntax:

hu?

OK Response:
+<historical_logged_users_count>

Formats of Positive Response Value:
historical_logged_users_count: int

Error Response:
Only defaults

Considerations:
This command only considers the amount of successful logins that were made to the POP3 server.

LIST CAPABILITIES(cap?)

Already defined in the AUTHENTICATION state section

QUIT(q)

Already defined in the AUTHENTICATION state section

5 Protocol Extension

This protocol can be extended in the future by adding more commands to any of the states. All new commands must follow the syntax defined in section 2.1 of this document. These must be listed in the response of the ‘cap?’ command, enabling a client to identify if the server implements an extended version or not.

As mentioned earlier, this protocol sets the restriction of two arguments or less for each command. If more than two arguments are needed to implement a command, the additional arguments may be concatenated to one of the first two, using a special character of choice (other than the space character). However, the implementation of such commands is not recommended in order to keep the protocol simple.

6 PEEP Commands Summary

Command (argc)	Valid States	Positive Response Format	Special Negative Responses
a (2)	AUTHENTICATION	none	-2
cap? (0)	BOTH	multi-line string	
q (0)	BOTH	none	

u+ (2)	AUTHENTICATED	none	-3
u- (1)	AUTHENTICATED	none	-4
u? (0)	AUTHENTICATED	multi-line string	
c= (1)	AUTHENTICATED	none	-5
c? (0)	AUTHENTICATED	integer	
m? (0)	AUTHENTICATED	string	
m= (1)	AUTHENTICATED	none	-6
t? (0)	AUTHENTICATED	integer	
t= (1)	AUTHENTICATED	none	-7
rb? (0)	AUTHENTICATED	integer	
re? (0)	AUTHENTICATED	integer	
xe? (0)	AUTHENTICATED	integer	
cc? (0)	AUTHENTICATED	integer	
cu? (0)	AUTHENTICATED	multi-line string	
hc? (0)	AUTHENTICATED	integer	
hu? (0)	AUTHENTICATED	integer	

4 Authors

- Federico Shih
- Mauro Leandro Báez
- Franco David Rupnik
- Matías Manzur