Network Working Group
Request for Comments: 3656
Category: Experimental

R. Siemborski
Carnegie Mellon University
December 2003

The Mailbox Update (MUPDATE) Distributed Mailbox Database Protocol

Status of this Memo

Copyright Notice

Abstract

As the demand for high-performance mail delivery agents increases, it becomes apparent that single-machine solutions are inadequate to the task, both because of capacity limits and that the failure of the single machine means a loss of mail delivery for all users. It is preferable to allow many machines to share the responsibility of mail delivery.

The Mailbox Update (MUPDATE) protocol allows a group of Internet Message Access Protocol (IMAP) or Post Office Protocol - Version 3 (POP3) servers to function with a unified mailbox namespace. This document is intended to serve as a reference guide to that protocol.

Table of Contents

1. Introduction

In order to support an architecture where there are multiple [IMAP, POP3] servers sharing a common mailbox database, it is necessary to be able to provide atomic mailbox operations, as well as offer sufficient guarantees about database consistency.

The primary goal of the MUPDATE protocol is to be simple to implement yet allow for database consistency between participants.

The key words "MUST, "MUST NOT", "REQUIRED", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are to be interpreted as defined in BCP 14, RFC 2119 [KEYWORDS].

In examples, "C:" and "S:" indicate lines sent by the client and server respectively.

2. Protocol Overview

The MUPDATE protocol assumes a reliable data stream such as a TCP network connection. IANA has registered port 3905 with a short name of "mupdate" for this purpose.

In the current implementation of the MUPDATE protocol there are three types of participants: a single master server, slave (or replica) servers, and clients. The master server maintains an authoritative copy of the mailbox database. Slave servers connect to the MUPDATE master server as clients, and function as replicas from the point of view of end clients. End clients may connect to either the master or any slave and perform searches against the database, however operations that change the database can only be performed against the master. For the purposes of protocol discussion we will consider a slave’s connection to the master identical to that of any other client.

After connection, all commands from a client to server must have an associated unique tag which is an alphanumeric string. Commands MAY be pipelined from the client to the server (that is, the client need not wait for the response before sending the next command). The server MUST execute the commands in the order they were received, however.

If the server supports an inactivity login timeout, it MUST be at least 15 minutes.

MUPDATE uses data formats similar to those used in [ACAP]. That is, atoms and strings. All commands and tags in the protocol are transmitted as atoms. All other data is considered to a string, and must be quoted or transmitted as a literal.

Outside of a literal, both clients and servers MUST support line lengths of at least 1024 octets (including the trailing CR and LF characters). If a line of a longer length must be transmitted, implementations MUST make use of literals to do so.

2.1. Atoms

An atom consists of one or more alphanumeric characters. Atoms MUST be less than 15 octets in length.

2.2. Strings

As in [ACAP], a string may be either literal or a quoted string. A literal is a sequence of zero or more octets (including CR and LF), prefix-quoted with an octet count in the form of an open brace ("{"), the number of octets, an optional plus sign to indicate that the data follows immediately (a non-synchronized literal), a close brace ("}"), and a CRLF sequence. If the plus sign is omitted (a synchronized literal), then the receiving side MUST send a "+ go ahead" response, and the sending side MUST wait for this response. Servers MUST support literals of atleast 4096 octets.

Strings that are sent from server to client SHOULD NOT be in the synchronized literal format.

A quoted string is a sequence of zero or more 7-bit characters, excluding CR, LF, and the double quote (<">), with double quote characters at each end.

The empty string is represented as either "" (a quoted string with zero characters between double quotes) or as {0} followed by CRLF (a literal with an octet count of 0).

3. Server Responses

Every client command in the MUPDATE protocol may receive one or more tagged responses from the server. Each response is preceded by the same tag as the command that elicited the response from the server.

3.1. Response: OK

A tagged OK response indicates that the operation completed successfully. There is a mandatory implementation-defined string after the OK response. This response also indicates the beginning of the streaming update mode when given in response to an UPDATE command.

Example:

```
C: N01 NOOP
S: N01 OK "NOOP Complete"
```

3.2. Response: NO

A tagged NO response indicates that the operation was explicitly denied by the server or otherwise failed. There is a mandatory implementation-defined string after the NO response that SHOULD explain the reason for denial.

Example:

```
C: A01 AUTHENTICATE "PLAIN"
S: A01 NO "PLAIN is not a supported SASL mechanism"
```

3.3. Response: BAD

A tagged BAD response indicates that the command from the client could not be parsed or understood. There is a mandatory implementation-defined string after the BAD response to provide additional information about the error. Note that untagged BAD responses are allowed if it is unclear what the tag for a given command is (for example, if a blank line is received by the mupdate server, it can generate an untagged BAD response). In the case of an untagged response, the tag should be replaced with a "*".

Example:

```
C: C01 SELECT "INBOX"
S: C01 BAD "This is not an IMAP server"
C:
S: * BAD "Need Command"
```

3.4. Response: BYE

A tagged BYE response indicates that the server has decided to close the connection. There is a mandatory implementation-defined string after the BYE response that SHOULD explain the reason for closing the connection. The server MUST close the connection immediately after transmitting the BYE response.

Example:

```
C: L01 LOGOUT
S: L01 BYE "User Logged Out"
```

3.5. Response: RESERVE

A tagged RESERVE response may only be given in response to a FIND, LIST, or UPDATE command. It includes two parameters: the name of the mailbox that is being reserved (in mUTF-7 encoding, as specified in [IMAP]) and a location string whose contents is defined by the clients that are using the database, though it is RECOMMENDED that the format of this string be the hostname of the server which is storing the mailbox.

This response indicates that the given name is no longer available in the namespace, though it does not indicate that the given mailbox is available to clients at the current time.

Example:

```
S: U01 RESERVE "internet.bugtraq" "mail2.example.org"
```

3.6. Response: MAILBOX

A tagged MAILBOX response may only be given in response to a FIND, LIST, or UPDATE command. It includes three parameters: the name of the mailbox, a location string (as with RESERVE), and a client-defined string that specifies the IMAP ACL [IMAP-ACL] of the mailbox. This message indicates that the given mailbox is ready to be accessed by clients.

Example:

```
S: U01 MAILBOX "internet.bugtraq" "mail2.example.org" "anyone rls"
```

3.7. Response: DELETE

A tagged DELETE response may only be given in response to an UPDATE command, and MUST NOT be given before the OK response to the UPDATE command is given. It contains a single parameter, that of the mailbox that should be deleted from the slave's database. This response indicates that the given mailbox no longer exists in the namespace of the database, and may be given for any mailbox name, active, reserved, or nonexistent. (Though implementations SHOULD NOT issue DELETE responses for nonexistent mailboxes).

Example:

```
S: U01 DELETE "user.rjs3.sent-mail-jan-2002"
```

3.8. Server Capability Response

Upon connection of the client to the server, and directly following a successful STARTTLS command, the server MUST issue a capabilities banner, of the following format:

The banner MUST contain a line that begins with "* AUTH" and contain a space-separated list of SASL mechanisms that the server will accept for authentication. The mechanism names are transmitted as atoms. Servers MAY advertise no available mechanisms (to indicate that STARTTLS must be completed before authentication may occur). If STARTTLS is not supported by the server, then the line MUST contain at least one mechanism.

If the banner is being issued without a TLS layer, and the server supports the STARTTLS command, the banner MUST contain the line "* STARTTLS". If the banner is being issued under a TLS layer (or the server does not support STARTTLS), the banner MUST NOT contain this line.

The last line of the banner MUST start with "* OK MUPDATE" and be followed by four strings: the server's hostname, an implementation-defined string giving the name of the implementation, an implementation-defined string giving the version of the implementation, and a string that indicates if the server is a master or a slave. The master/slave indication MUST be either "(master)" or an MUPDATE URL that defines where the master can be contacted.

Any unrecognized responses before the "* OK MUPDATE" response MUST be ignored by the client.

Example:

```
S: * AUTH KERBEROS_V4 GSSAPI
S: * STARTTLS
S: * OK MUPDATE "mupdate.example.org" "Cyrus" "v2.1.2" "(master)"
```

4. Client Commands

The following are valid commands that a client may send to the MUPDATE server: AUTHENTICATE, ACTIVATE, DEACTIVATE, DELETE, FIND, LIST, LOGOUT, NOOP, RESERVE, STARTTLS, and UPDATE.

Before a successful AUTHENTICATE command has occurred, the server MUST NOT accept any commands except for AUTHENTICATE, STARTTLS, and LOGOUT (and SHOULD reply with a NO response for all other commands).

4.1. Command: ACTIVATE

The ACTIVATE command has 3 parameters: the mailbox name, its location, and its ACL. This command MUST NOT not be issued to a slave server.

This command can also be used to update the ACL or location information of a mailbox. Note that it is not a requirement for a mailbox to be reserved (or even exist in the database) for an ACTIVATE command to succeed, implementations MUST allow this behavior as it facilitates synchronization of the database with the current state of the mailboxes.

4.2. Command: AUTHENTICATE

The AUTHENTICATE command initiates a [SASL] negotiation session between the client and the server. It has two parameters. The first parameter is mandatory, and is a string indicating the desired [SASL] mechanism. The second is a string containing an optional BASE64 encoded (as defined in section 6.8 of [MIME]) client first send.

All of the remaining SASL blobs that are sent MUST be sent across the wire must be in BASE64 encoded format, and followed by a CR and LF combination. They MUST NOT be encoded as strings.

Clients may cancel authentication by sending a * followed by a CR and LF.

The [SASL] service name for the MUPDATE protocol is "mupdate". Implementations are REQUIRED to implement the GSSAPI [SASL] mechanism, though they SHOULD implement as many mechanisms as possible.

If a security layer is negotiated, it should be used directly following the CR and LF combination at the end of the server’s OK response (i.e., beginning with the client’s next command) Only one successful AUTHENTICATE command may be issued per session.

4.3. Command: DEACTIVATE

The DEACTIVATE command takes two parameters, the mailbox name and location data. The mailbox MUST already exist and be activated on the MUPDATE server. If the server responds OK, then the mailbox name has been moved to the RESERVE state. If the server responds NO, then the mailbox name has not been moved (for example, the mailbox was not already active). Any ACL information that is known about the mailbox MAY be lost when a DEACTIVATE succeeds. This command MUST NOT be issued to a slave.

Example:

```
C: A01 DEACTIVATE "user.rjs3.new" "mail3.example.org!u4"
S: A01 OK "Mailbox Reserved."
```

4.4. Command: DELETE

The DELETE command takes only a single parameter, the mailbox name to be removed from the database’s namespace. The server SHOULD give a NO response if the mailbox does not exist. This command MUST NOT be issued to a slave server.

4.5. Command: FIND

The FIND command takes a single parameter, a mailbox name. The server then responds with the current record for the given mailbox, if any, and an OK response.

Example (mailbox does not exist):

```
C: F01 FIND "user.rjs3.xyzzy"
S: F01 OK "Search Complete"
```

Example (mailbox is reserved):

```
C: F01 FIND "user.rjs3"
S: F01 RESERVE "user.rjs3" "mail4.example.org"
S: F01 OK "Search Complete"
```

4.6. Command: LIST

The LIST command is similar to running FIND across the entire database. The LIST command takes a single optional parameter, which is a prefix to try to match against the location field of the records. Without the parameter, LIST returns every record in the database.

For each mailbox that matches, either a MAILBOX or a RESERVE response (as applicable) is sent to the client. When all responses are complete, an OK response is issued.

Example:

```
C: L01 LIST
S: L01 RESERVE "user.rjs3" "mail4.example.org!u2"
S: L01 MAILBOX "user.leg" "mail2.example.org!u1" "leg lrswipcda"
S: L01 OK "List Complete"
C: L02 LIST "mail4.example.org!"
S: L02 RESERVE "user.rjs3" "mail4.example.org!u2"
S: L02 OK "List Complete"
```

4.7. Command: LOGOUT

The LOGOUT command tells the server to close the connection. Its only valid response is the BYE response. The LOGOUT command takes no parameters.

4.8. Command: NOOP

The NOOP command takes no parameters. Provided the client is authenticated, its only acceptable response is an OK. Any idle timeouts that the server may have on the connection SHOULD be reset upon receipt of this command.

If this command is issued after an UPDATE command has been issued, then the OK response also indicates that all pending database updates have been sent to the client. That is, the slave can guarantee that its local database is up to date as of a certain time by issuing a NOOP and waiting for the OK. The OK MUST NOT return until all updates that were pending at the time of the NOOP have been sent.

4.9. Command: RESERVE

The RESERVE command takes two parameters (just like the RESERVE response), the mailbox name to reserve and location data. If the server responds OK, then the mailbox name has been reserved. If the server responds NO, then the mailbox name has not been reserved (for example, another server has reserved it already). This command MUST NOT be issued to a slave.

The typical sequence for mailbox creation is:

```
C: R01 RESERVE "user.rjs3.new" "mail3.example.org!u4"
S: R01 OK "Mailbox Reserved."
<client does local mailbox create operations>
C: A01 ACTIVATE "user.rjs3.new" "mail3.example.org!u4" "rjs3 lrswipcda"
S: A01 OK "Mailbox Activated."
```

4.10. Command: STARTTLS

The STARTTLS command requests the commencement of a [TLS] negotiation. The negotiation begins immediately after the CRLF in the OK response. After a client issues a STARTTLS command, it MUST NOT issue further commands until a server response is seen and the [TLS] negotiation is complete.

The STARTTLS command is only valid in non-authenticated state. The server remains in non-authenticated state, even if client credentials are supplied during the [TLS] negotiation. The [SASL] EXTERNAL mechanism MAY be used to authenticate once [TLS] client credentials are successfully exchanged. Note that servers are not required to support the EXTERNAL mechanism.

After the [TLS] layer is established, the server MUST re-issue the initial response banner (see Section 3.8). This is necessary to protect against man-in-the-middle attacks which alter the capabilities list prior to STARTTLS, as well as to advertise any new SASL mechanisms (or other capabilities) that may be available under the layer. The client MUST discard cached capability information and replace it with the new information.

After the a successful STARTTLS command, the server SHOULD return a NO response to additional STARTTLS commands.

Servers MAY choose to not implement STARTTLS. In this case, they MUST NOT advertise STARTTLS in their capabilities banner, and SHOULD return a BAD response to the STARTTLS command, if it is issued.

Example:

```
C: S01 STARTTLS
S: S01 OK "Begin TLS negotiation now"
<TLS negotiation, further commands are under TLS layer>
S: * AUTH KERBEROS_V4 GSSAPI PLAIN
S: * OK MUPDATE "mupdate.example.org" "Cyrus" "v2.1.2" "(master)"
```

4.11. Command: UPDATE

The UPDATE command is how a slave initializes an update stream from the master (though it is also valid to issue this command to a slave). In response to the command, the server returns a list of all mailboxes in its database (the same results as a parameterless LIST command) followed by an OK response. From this point forward, whenever an update occurs to the master database, it MUST stream the update to the slave within 30 seconds. That is, it will send RESERVE, MAILBOX, or DELETE responses as they are applicable.

After a client has issued an UPDATE command, it may only issue NOOP and LOGOUT commands for the remainder of the session.

Example:

```
C: U01 UPDATE
S: U01 MAILBOX "user.leg" "mail2.example.org!u1" "leg lrswipcda"
S: U01 MAILBOX "user.rjs3" "mail3.example.org!u4" "rjs3 lrswipcda"
S: U01 RESERVE "internet.bugtraq" "mail1.example.org!u5" "anyone lrs"
S: U01 OK "Streaming Begins"
<some time goes by, and another client creates a new mailbox>
S: U01 RESERVE "user.leg.new" "mail2.example.org!u1"
<some more time passes, and the create succeeds>
S: U01 MAILBOX "user.leg.new" "mail2.example.org!u1" "leg lrswipcda"
<much more time passes, and the slave decides to send a NOOP to reset
its inactivity timer>
C: N01 NOOP
S: U01 DELETE "user.leg.new"
S: N01 OK "NOOP Complete"
```

5. MUPDATE Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [ABNF]. This uses the ABNF core rules as specified in Appendix A of [ABNF].

Except as noted otherwise, all alphabetic characters are case-insensitive. The use of upper or lower case characters to define token strings is for editorial clarity only. Implementations MUST accept these strings in a case-insensitive fashion.

Note that this specification also uses some terminals from section 8 of [ACAP].

```
   cmd-activate = "ACTIVATE" SP string SP string SP string

   cmd-authenticate = "AUTHENTICATE" SP sasl-mech [ SP string ]
   cmd-delete = "DELETE" SP string

   cmd-find = "FIND" SP string

   cmd-list = "LIST" [ SP string ]

   cmd-logout = "LOGOUT"

   cmd-noop = "NOOP"

   cmd-reserve = "RESERVE" SP string SP string

   cmd-starttls = "STARTTLS"

   cmd-update = "UPDATE"

   command = tag SP command-type CRLF

   command-type = cmd-activate / cmd-authenticate / cmd-delete /
                  cmd-find / cmd-list / cmd-logout / cmd-noop /
                  cmd-reserve / cmd-starttls / cmd-update

   response = tag SP response-type CRLF

   response-type = rsp-ok / rsp-no / rsp-bad / rsp-bye / rsp-mailbox /
                   rsp-reserve / rsp-delete

   rsp-bad = "BAD" SP string

   rsp-bye = "BYE" SP string

   rsp-mailbox = "MAILBOX" SP string SP string SP string

   rsp-no = "NO" SP string

   rsp-ok = "OK" SP string

   rsp-reserve = "RESERVE" SP string SP string

   rsp-delete = "DELETE" SP string

   sasl-mech = 1*ATOM-CHAR
      ; ATOM-CHAR is defined in [ACAP]

   string = quoted / literal
      ; quoted and literal are defined in [ACAP]
   tag = 1*ATOM-CHAR
      ; ATOM-CHAR is defined in [ACAP]
```

6. MUPDATE URL Scheme

This document defines the a URL scheme for the purposes of referencing MUPDATE resources, according to the requirements in [RFC2717]. This includes both MUPDATE servers as a whole, along with individual mailbox entries on a given MUPDATE server.

There is no MIME type associated with these resources. It is intended that a URL consumer would either retrieve the MUPDATE record in question, or simply connect to the MUPDATE server running on the specified host. Note that the consumer will need to have authentication credentials for the specified host.

The MUPDATE URL scheme is similar to the IMAP URL scheme [IMAP-URL]. However, it only takes one of two possible forms:

```
      mupdate://<iserver>/
      mupdate://<iserver>/<mailbox>
```

The first form refers to a MUPDATE server as a whole, the second form indicates both the server and a mailbox to run a FIND against once authenticated to the server. Note that part of <iserver> may include username and authentication information along with a hostname and port.

6.1. MUPDATE URL Scheme Registration Form

URL scheme name: "mupdate"

URL scheme syntax:

This defines the MUPDATE URL Scheme in [ABNF]. Terminals from the BNF of IMAP URLs [IMAP-URL] are also used.

```
         mupdateurl = "mupdate://" iserver "/" [ enc_mailbox ]
            ; iserver and enc_mailbox are as defined in [IMAP-URL]
```

Character encoding considerations:

Identical to those described in [IMAP-URL] for the appropriate terminals.

Intended Usage:

The form of the URL without an associated mailbox is intended to designate a MUPDATE server only. If a mailbox name is included in the URL, then the consumer is expected to execute a FIND command for that mailbox on the specified server.

Applications and/or protocols which use this URL scheme name:

The protocol described in this document.

Interoperability Considerations:

None.

Security Considerations:

Users of the MUPDATE URL Scheme should review the security considerations that are discussed in [IMAP-URL]. In particular, the consequences of including authentication mechanism information in a URL should be reviewed.

Relevant Publications:

This document and [IMAP-URL].

Author, Change Controller, and Contact for Further Information:

Author of this document.

7. Security Considerations

While no unauthenticated users may make modifications or even perform searches on the database, it is important to note that this specification assumes no protections of any type for authenticated users.

All authenticated users have complete access to the database. For this reason it is important to ensure that accounts that are making use of the database are well secured.

A more secure deployment might have all read only access go through a slave, and only have accounts which need write access use the master. This has the disadvantage of a marginally longer time for updates to reach the clients.

The protocol assumes that all authenticated users are cooperating to maintain atomic operations. Therefore, all new mailboxes SHOULD be RESERVEd before they are ACTIVATEd, despite the fact that the protocol does not require this, and it is therefore possible for a set of participants which do not obey the provided locking to create an inconsistent database. RESERVEing the mailbox first is not required to perform an activate because this behavior simplifies synchronization with the actual location of the mailboxes.

8. IANA Considerations

The IANA has assigned TCP port number 3905 to "mupdate".

The IANA has registered a URL scheme for the MUPDATE protocol, as defined in section 6.1 of this document.

IANA has registered a GSSAPI service name of "mupdate" for the MUPDATE protocol in the registry maintained at:

http://www.iana.org/assignments/gssapi-service-names

9. Intellectual Property Rights

10. References

10.1. Normative References

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[IMAP] Crispin, M., "Internet Message Access Protocol - Version 4", RFC 3501, March 2003.

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[MIME] Freed, N. and N. Bornstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, November 1996.

[IMAP-ACL] Myers, J., "IMAP4 ACL extension", RFC 2086, January 1997.

[SASL] Myers, J., "Simple Authentication and Security Layer (SASL)", RFC 2222, October 1997.

[IMAP-URL] Newman, C., "IMAP URL Scheme", RFC 2192, September 1997.

[ACAP] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[TLS] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

10.2. Informative References

[POP3] Myers, J. and M. Rose, "Post Office Protocol - Version 3", STD 53, RFC 1939, May 1996.

[RFC2717] Petke, R. and I. King, "Registration Procedures for URL Scheme Names", BCP 35, RFC 2717, November 1999.

11. Acknowledgments

Lawrence Greenfield and Ken Murchison, for a great deal of input on both the protocol and the text of the documents.

12. Author's Address

Robert Siemborski
Carnegie Mellon, Andrew Systems Group
Cyert Hall 207
5000 Forbes Avenue
Pittsburgh, PA 15213

Phone: (412) 268-7456
EMail: rjs3+@andrew.cmu.edu

13. Full Copyright Statement

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.