INTERNET MESSAGE ACCESS PROTOCOL - MULTIAPPEND EXTENSION

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

A revised version of this draft document will be submitted to the RFC editor as a Proposed Standard for the Internet Community.

Discussion and suggestions for improvement are requested, and should be sent to ietf-imapext@IMC.ORG. This document will expire before 10 July 2000. Distribution of this memo is unlimited.

Abstract

This document describes the multiappending extension to the [IMAP] protocol. This extension provides substantial performance improvements for IMAP clients which upload multiple messages at a time to a mailbox on the server.

A server which supports this extension indicates this with a capability name of "MULTIAPPEND".

Crispin
Introduction

The MULTIAPPEND extension permits uploading of multiple messages with a single command. When used in conjunction with the [LITERAL+] extension, the entire upload is accomplished in a single command/response round trip.

A MULTIAPPEND APPEND operation is atomic; either all messages are successfully appended, or no messages are appended.

In the base IMAP specification, each message must be appended in a separate command, and there is no mechanism to "unappend" messages if an error occurs while appending. Also, some mail stores may require an expensive "open/lock + sync/unlock/close" operation as part of appending; this can be quite expensive if it must be done on a per-message basis.

If the server supports both LITERAL+ and pipelining but not MULTIAPPEND, it may be possible to get some of the performance advantages of MULTIAPPEND by doing a pipelined "batch" append. However, it will not work as well as MULTIAPPEND for the following reasons:

1) Multiple APPEND commands, even as part of a pipelined batch, are non-atomic by definition. There is no way to revert the mailbox to the state before the batch append in the event of an error.

2) It may not be feasible for the server to coalesce pipelined APPEND operations so as to avoid the "open/lock + sync/unlock/close" overhead described above. In any case, such coalescing would be timing dependent and thus potentially unreliable. In particular, with traditional UNIX mailbox files, it is assumed that a lock is held only for a single atomic operation, and many applications disregard any lock that is older than 5 minutes.

3) If an error occurs, depending upon the nature of the error, it is possible for additional messages to be appended after the error. For example, the user wants to append 5 messages, but a disk quota error occurs with the third message because of its size. However, the fourth and fifth messages have already been sent in the pipeline, so the mailbox ends up with the first, second, fourth, and fifth messages of the batch appended.
Extension to IMAP4rev1 Base Protocol Commands

6.3.11. APPEND Command

Arguments: mailbox name
one or more messages to upload, specified as:
   OPTIONAL flag parenthesized list
   OPTIONAL date/time string
   message literal

Data: no specific responses for this command

Result: OK - append completed
NO - append error: can’t append to that mailbox, error
   in flags or date/time or message text
BAD - command unknown or arguments invalid

The APPEND command appends the literal arguments as new messages
to the end of the specified destination mailbox. This argument
SHOULD be in the format of an [RFC-822] message. 8-bit characters
are permitted in the message. A server implementation that is
unable to preserve 8-bit data properly MUST be able to reversibly
convert 8-bit APPEND data to 7-bit using a [MIME-IMB] content
transfer encoding.

Note: There MAY be exceptions, e.g. draft messages, in
which required [RFC-822] header lines are omitted in the
message literal argument to APPEND. The full
implications of doing so MUST be understood and
carefully weighed.

If a flag parenthesized list is specified, the flags SHOULD be set
in the resulting message; otherwise, the flag list of the
resulting message is set empty by default.

If a date-time is specified, the internal date SHOULD be set in
the resulting message; otherwise, the internal date of the
resulting message is set to the current date and time by default.

If the append is unsuccessful for any reason, the mailbox MUST be
restored to its state before the APPEND attempt; no partial
appending is permitted. The server MAY return an error before
processing all the message arguments.

If the destination mailbox does not exist, a server MUST return an
error, and MUST NOT automatically create the mailbox. Unless it
is certain that the destination mailbox can not be created, the
server MUST send the response code "[TRYCREATE]" as the prefix of
the text of the tagged NO response. This gives a hint to the
client that it can attempt a CREATE command and retry the APPEND
if the CREATE is successful.

If the mailbox is currently selected, the normal new message
actions SHOULD occur. Specifically, the server SHOULD notify the
client immediately via an untagged EXISTS response. If the server
does not do so, the client MAY issue a NOOP command (or failing
that, a CHECK command) after one or more APPEND commands.

Example:    C: A003 APPEND saved-messages (\Seen) {310}
             S: + Ready for literal data
             C: Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
             C: From: Fred Foobar <foobar@Blurdybloop.COM>
             C: Subject: afternoon meeting
             C: To: mooch@owatagu.siam.edu
             C: Message-Id: <B27397-0100000@Blurdybloop.COM>
             C: MIME-Version: 1.0
             C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
             C:
             C: Hello Joe, do you think we can meet at 3:30 tomorrow?
             C:  (\Seen) " 7-Feb-1994 22:43:04 -0800" (286)
             S: + Ready for literal data
             C: Date: Mon, 7 Feb 1994 22:43:04 -0800 (PST)
             C: From: Joe Mooch <mooch@OWaTaGu.siam.EDU>
             C: Subject: Re: afternoon meeting
             C: To: foobar@blurdybloop.com
             C: Message-Id: <a0434793874930@OWaTaGu.siam.EDU>
             C: MIME-Version: 1.0
             C: Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
             C:
             C: 3:30 is fine with me.
             C:
             S: A003 OK APPEND completed
             C: A004 APPEND bogusname (\Flagged) {1023}
             S: A004 NO [TRYCREATE] No such mailbox as bogusname
             C: A005 APPEND test (\Flagged) {99}
             S: + Ready for literal data
             C: Date: Mon, 7 Feb 2000 22:43:04 -0800 (PST)
             C: From: Fred Foobar <fred@foobar.com>
             C: Subject: hmm...
             C:  {35403}
             S: A005 NO APPEND failed: Disk quota exceeded

Note: The APPEND command is not used for message delivery,
because it does not provide a mechanism to transfer [SMTP]
envelope information.
Modification to IMAP4rev1 Base Protocol Formal Syntax

append = "APPEND" SPACE mailbox 1*append_message
append-message = [SPACE flag_list] [SPACE date_time] SPACE literal

MULTIAPPEND Interaction with UIDPLUS Extension

Servers which support both MULTIAPPEND and [UIDPLUS] will have the
"resp-code-apnd" rule modified as follows:

resp-code-apnd = "APPENDUID" SPACE nz_number 1*(SPACE uniqueid)

That is, the APPENDUID response code returns as many UIDs as there
were messages appended in the multiple append.

Security Considerations

Security issues are not discussed in this memo.

References


Author’s Address

Mark R. Crispin
Networks and Distributed Computing
University of Washington
4545 15th Avenue NE
Seattle, WA 98105-4527

Phone: (206) 543-5762

EMail: MRC@CAC.Washington.EDU