Network Working Group
Request for Comments: 4469
Updates: 3501, 3502
Category: Standards Track

P. Resnick
QUALCOMM Incorporated

April 2006

Internet Message Access Protocol (IMAP) CATENATE Extension

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

The CATENATE extension to the Internet Message Access Protocol (IMAP) extends the APPEND command to allow clients to create messages on the IMAP server that may contain a combination of new data along with parts of (or entire) messages already on the server. Using this extension, the client can catenate parts of an already existing message onto a new message without having to first download the data and then upload it back to the server.

1. Introduction

The CATENATE extension to the Internet Message Access Protocol (IMAP) [1] allows the client to create a message on the server that can include the text of messages (or parts of messages) that already exist on the server without having to FETCH them and APPEND them back to the server. The CATENATE extension extends the APPEND command so that, instead of a single message literal, the command can take as arguments any combination of message literals (as described in IMAP [1]) and message URLs (as described in the IMAP URL Scheme [2] specification). The server takes all the pieces and catenates them into the output message. The CATENATE extension can also coexist with the MULTIAPPEND extension [3] to APPEND multiple messages in a single command.

There are some obvious uses for the CATENATE extension. The motivating use case was to provide a way for a resource-constrained client to compose a message for subsequent submission that contains data that already exists in that client's IMAP store. Because the client does not have to download and re-upload potentially large message parts, bandwidth and processing limitations do not have as much impact. In addition, since the client can create a message in its own IMAP store, the command also addresses the desire of the client to archive a copy of a sent message without having to upload the message twice. (Mechanisms for sending the message are outside the scope of this document.)

The extended APPEND command can also be used to copy parts of a message to another mailbox for archival purposes while getting rid of undesired parts. In environments where server storage is limited, a client could get rid of large message parts by copying over only the necessary parts and then deleting the original message. The mechanism could also be used to add data to a message (such as prepending message header fields) or to include other data by making a copy of the original and catenating the new data.

2. The CATENATE Capability

A server that supports this extension returns "CATENATE" as one of the responses to the CAPABILITY command.

3. The APPEND Command

```
Arguments:  mailbox name
            (The following can be repeated in the presence of the
            MULTIAPPEND extension [3])
            OPTIONAL flag parenthesized list
            OPTIONAL date/time string
            a single message literal or one or more message parts to
            catenate, specified as:
                        message literal
                        or
                        message (or message part) URL

Responses:  OPTIONAL NO responses: BADURL, TOOBIG

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

The APPEND command concatenates all the message parts and appends them as a new message to the end of the specified mailbox. The parenthesized flag list and date/time string set the flags and the internal date, respectively, as described in IMAP [1]. The subsequent command parameters specify the message parts that are appended sequentially to the output message.

If the original form of APPEND is used, a message literal follows the optional flag list and date/time string, which is appended as described in IMAP [1]. If the extended form is used, "CATENATE" and a parenthesized list of message literals and message URLs follows, each of which is appended to the new message. If a message literal is specified (indicated by "TEXT"), the octets following the count are appended. If a message URL is specified (indicated by "URL"), the octets of the body part pointed to by that URL are appended, as if the literal returned in a FETCH BODY response were put in place of the message part specifier. The APPEND command does not cause the \Seen flag to be set for any catenated body part. The APPEND command does not change the selected mailbox.

In the extended APPEND command, the string following "URL" is an IMAP URL [2] and is interpreted according to the rules of [2]. The present document only describes the behavior of the command using IMAP URLs that refer to specific messages or message parts on the current IMAP server from the current authenticated IMAP session. Because of that, only relative IMAP message or message part URLs (i.e., those having no scheme or <iserver>) are used. The base URL

for evaluating the relative URL is considered "imap://user@server/", where "user" is the user name of the currently authenticated user and "server" is the domain name of the current server. When in the selected state, the base URL is considered "imap://user@server/mailbox", where "mailbox" is the encoded name of the currently selected mailbox. Additionally, since the APPEND command is valid in the authenticated state of an IMAP session, no further LOGIN or AUTHENTICATE command is performed for URLs specified in the extended APPEND command.

> Note: Use of an absolute IMAP URL or any URL that refers to anything other than a message or message part from the current authenticated IMAP session is outside the scope of this document and would require an extension to this specification, and a server implementing only this specification would return NO to such a request.

The client is responsible for making sure that the catenated message is in the format of an Internet Message Format (RFC 2822) [4] or Multipurpose Internet Mail Extension (MIME) [5] message. In particular, when a URL is catenated, the server copies octets, unchanged, from the indicated message or message part to the catenated message. It does no data conversion (e.g., MIME transfer encodings) nor any verification that the data is appropriate for the MIME part of the message into which it is inserted. The client is also responsible for inserting appropriate MIME boundaries between body parts, and writing MIME Content-Type and Content-Transfer-Encoding lines as needed in the appropriate places.

Responses behave just as the original APPEND command described in IMAP [1]. If the server implements the IMAP UIDPLUS extension [6], it will also return an APPENDUID response code in the tagged OK response. Two response codes are provided in Section 4 that can be used in the tagged NO response if the APPEND command fails.

4. Response Codes

When a APPEND command fails, it may return a response code that describes a reason for the failure.

4.1. BADURL Response

The BADURL response code is returned if the APPEND fails to process one of the specified URLs. Possible reasons for this are bad URL syntax, unrecognized URL schema, invalid message UID, or invalid body part. The BADURL response code contains the first URL specified as a parameter to the APPEND command that has caused the operation to fail.

4.2. TOOBIG Response

The TOOBIG response code is returned if the resulting message will exceed the 4-GB IMAP message limit. This might happen, for example, if the client specifies 3 URLs for 2-GB messages. Note that even if the server doesn't return TOOBIG, it still has to be defensive against misbehaving or malicious clients that try to construct a message over the 4-GB limit. The server may also wish to return the TOOBIG response code if the resulting message exceeds a server-specific message size limit.

5. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) [7] notation. Elements not defined here can be found in the formal syntax of the ABNF [7], IMAP [1], and IMAP ABNF extensions [8] specifications. Note that capability and resp-text-code are extended from the IMAP [1] specification and append-data is extended from the IMAP ABNF extensions [8] specification.

```
append-data =/ "CATENATE" SP "(" cat-part *(SP cat-part) ")"

cat-part = text-literal / url

text-literal = "TEXT" SP literal

url = "URL" SP astring

resp-text-code =/ toobig-response-code / badurl-response-code

toobig-response-code = "TOOBIG"

badurl-response-code = "BADURL" SP url-resp-text

url-resp-text = 1*(%x01-09 /
                   %x0B-0C /
                   %x0E-5B /
                   %x5D-FE) ; Any TEXT-CHAR except "]"

capability =/ "CATENATE"
```

The astring in the definition of url and the url-resp-text in the definition of badurl-response-code each contain an imapurl as defined by [2].

6. Acknowledgements

Thanks to the members of the LEMONADE working group for their input. Special thanks to Alexey Melnikov for the examples.

7. Security Considerations

The CATENATE extension does not raise any security considerations that are not present for the base protocol or in the use of IMAP URLs, and these issues are discussed in the IMAP [1] and IMAP URL [2] documents.

8. IANA Considerations

IMAP4 capabilities are registered by publishing a standards track or IESG approved experimental RFC. The registry is currently located at <http://www.iana.org/assignments/imap4-capabilities>. This document defines the CATENATE IMAP capability. The IANA has added this capability to the registry.

Appendix A. Examples

Lines not starting with "C: " or "S: " are continuations of the previous lines.

The original message in examples 1 and 2 below (UID = 20) has the following structure:

multipart/mixed MIME message with two body parts:

1. text/plain

2. application/x-zip-compressed

Example 1: The following example demonstrates how a CATENATE client can replace an attachment in a draft message, without the need to download it to the client and upload it back.

```
C: A003 APPEND Drafts (\Seen \Draft $MDNSent) CATENATE
 (URL "/Drafts;UIDVALIDITY=385759045/;UID=20/;section=HEADER"
 TEXT {42}
S: + Ready for literal data
C:
C: --------------030308070208000400050907
C:  URL "/Drafts;UIDVALIDITY=385759045/;UID=20/;section=1.MIME"
 URL "/Drafts;UIDVALIDITY=385759045/;UID=20/;section=1" TEXT {42}
S: + Ready for literal data
C:
C: --------------030308070208000400050907
C:  URL "/Drafts;UIDVALIDITY=385759045/;UID=30" TEXT {44}
S: + Ready for literal data
C:
C: --------------030308070208000400050907--
C: )
S: A003 OK catenate append completed
```

Example 2: The following example demonstrates how the CATENATE extension can be used to replace edited text in a draft message, as well as header fields for the top level message part (e.g., Subject has changed). The previous version of the draft is marked as \Deleted. Note that the server also supports the UIDPLUS extension, so the APPENDUID response code is returned in the successful OK response to the APPEND command.

```
C: A003 APPEND Drafts (\Seen \Draft $MDNSent) CATENATE (TEXT {738}
S: + Ready for literal data
C: Return-Path: <bar@example.org>
C: Received: from [127.0.0.2]
C:           by rufus.example.org via TCP (internal) with ESMTPA;
C:           Thu, 11 Nov 2004 16:57:07 +0000
C: Message-ID: <419399E1.6000505@example.org>
C: Date: Thu, 12 Nov 2004 16:57:05 +0000
C: From: Bob Ar <bar@example.org>
C: X-Accept-Language: en-us, en
C: MIME-Version: 1.0
C: To: foo@example.net
C: Subject: About our holiday trip
C: Content-Type: multipart/mixed;
C:               boundary="------------030308070208000400050907"
C:
C: --------------030308070208000400050907
C: Content-Type: text/plain; charset=us-ascii; format=flowed
C: Content-Transfer-Encoding: 7bit
C:
C: Our travel agent has sent the updated schedule.
C:
C: Cheers,
C: Bob
C: --------------030308070208000400050907
C:  URL "/Drafts;UIDVALIDITY=385759045/;UID=20/;Section=2.MIME"
 URL "/Drafts;UIDVALIDITY=385759045/;UID=20/;Section=2" TEXT {44}
S: + Ready for literal data
C:
C: --------------030308070208000400050907--
C: )
S: A003 OK [APPENDUID 385759045 45] append Completed
C: A004 UID STORE 20 +FLAGS.SILENT (\Deleted)
S: A004 OK STORE completed
```

Example 3: The following example demonstrates how the CATENATE extension can be used to strip attachments. Below, a PowerPoint attachment was replaced by a small text part explaining that the attachment was stripped.

```
C: A003 APPEND Drafts (\Seen \Draft $MDNSent) CATENATE
 (URL "/Drafts;UIDVALIDITY=385759045/;UID=21/;section=HEADER"
 TEXT {42}
S: + Ready for literal data
C:
C: --------------030308070208000400050903
C:  URL "/Drafts;UIDVALIDITY=385759045/;UID=21/;section=1.MIME"
 URL "/Drafts;UIDVALIDITY=385759045/;UID=21/;section=1" TEXT {255}
S: + Ready for literal data
C:
C: --------------030308070208000400050903
C: Content-type: text/plain; charset="us-ascii"
C: Content-transfer-encoding: 7bit
C:
C: This body part contained a Power Point presentation that was
C: deleted upon your request.
C: --------------030308070208000400050903--
C: )
S: A003 OK append Completed
```

Example 4: The following example demonstrates a failed APPEND command. The server returns the BADURL response code to indicate that one of the provided URLs is invalid. This example also demonstrates how the CATENATE extension can be used to construct a digest of several messages.

```
C: A003 APPEND Sent (\Seen $MDNSent) CATENATE (TEXT {541}
S: + Ready for literal data
C: Return-Path: <foo@example.org>
C: Received: from [127.0.0.2]
C:           by rufus.example.org via TCP (internal) with ESMTPA;
C:           Thu, 11 Nov 2004 16:57:07 +0000
C: Message-ID: <419399E1.6000505@example.org>
C: Date: Thu, 21 Nov 2004 16:57:05 +0000
C: From: Farren Oo <foo@example.org>
C: X-Accept-Language: en-us, en
C: MIME-Version: 1.0
C: To: bar@example.org
C: Subject: Digest of the mailing list for today
C: Content-Type: multipart/digest;
C:               boundary="------------030308070208000400050904"
C:
C: --------------030308070208000400050904
C:  URL "/INBOX;UIDVALIDITY=785799047/;UID=11467" TEXT {42}
S: + Ready for literal data
C:
C: --------------030308070208000400050904
C:  URL "/INBOX;UIDVALIDITY=785799047/;UID=113330/;section=1.5.9"
 TEXT {42}
S: + Ready for literal data
C:
C: --------------030308070208000400050904
C:  URL "/INBOX;UIDVALIDITY=785799047/;UID=11916" TEXT {44}
S: + Ready for literal data
C:
C: --------------030308070208000400050904--
C: )
S: A003 NO [BADURL "/INBOX;UIDVALIDITY=785799047/;UID=113330;
section=1.5.9"] CATENATE append has failed, one message expunged
```

Note that the server could have validated the URLs as they were received and therefore could have returned the tagged NO response with BADURL response-code in place of any continuation request after the URL was received.

9. Normative References

[1] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

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

[3] Crispin, M., "Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension", RFC 3502, March 2003.

[4] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

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

[6] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005.

[7] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[8] Melnikov, A. and C. Daboo, "Collected Extensions to IMAP4 ABNF", RFC 4466, April 2006.

Author's Address

Peter W. Resnick
QUALCOMM Incorporated
5775 Morehouse Drive
San Diego, CA 92121-1714
US

Phone: +1 858 651 4478
EMail: presnick@qualcomm.com
URI: http://www.qualcomm.com/~presnick/

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).