IMAP4 Extension: Message Preview Generation
draft-ietf-extra-imap-fetch-preview-01

Abstract

   This document specifies an IMAP protocol extension which allows a
   client to request that a server provide an abbreviated representation
   of a message that can be used by a client to provide a useful
   contextual preview of the message contents.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   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."

   This Internet-Draft will expire on July 26, 2019.

Copyright Notice

Table of Contents

1. Introduction

   Many modern mail clients display small extracts of the body text as
   an aid to allow a user to quickly decide whether they are interested
   in viewing the full message contents. Mail clients implementing the
   Internet Message Access Protocol [RFC3501] would benefit from a
   standardized, consistent way to generate these brief previews of
   messages.

   Generation of a preview on the server has several benefits. First,
   it allows consistent representation of previews across all clients.
   This standardized display can reduce user confusion when using
   multiple clients, as abbreviated message representations in clients
   will show identical message contents.

   Second, server-side preview generation is more efficient. A client-
   based algorithm needs to issue, at a minimum, a FETCH BODYSTRUCTURE
   command in order to determine which MIME [RFC2045] body part(s)
   should be represented in the preview. Subsequently, at least one
   FETCH BODY command may be needed to retrieve body data used in
   preview generation. These FETCH commands cannot be pipelined since
   the BODYSTRUCTURE query must be parsed on the client before the list
   of parts to be retrieved via the BODY command(s) can be determined.

Additionally, it may be difficult to predict the amount of body data
that must be retrieved to adequately represent the part via a
preview, therefore requiring inefficient fetching of excessive data
in order to account for this uncertainty. For example, a preview
algorithm to display data contained in a text/html [RFC2854] part
will likely strip the markup tags to obtain textual content.
However, without fetching the entire content of the part, there is no
way to guarantee that sufficient non-tag content will exist unless
either 1) the entire part is retrieved or 2) an additional partial
FETCH is executed when the client determines that it does not possess
sufficient data from a previous partial FETCH to display an adequate
representation of the preview.

Finally, server generation allows caching in a centralized location.
Using server generated previews allows global generation once per
message, and then cached indefinitely. Retrieval of message data may
be expensive within a server, for example, so a server can be
configured to reduce its storage retrieval load by pre-generating
preview data.

A server that supports the PREVIEW extension indicates this with one
or more capability names consisting of "PREVIEW=" followed by a
supported preview algorithm name. This format provides for future
upwards-compatible extensions and/or the ability to use locally-
defined preview algorithms.

2. Conventions Used In This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",
"SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and
"OPTIONAL" in this document are to be interpreted as described in BCP
14 [RFC2119] [RFC8174] when, and only when, they appear in all
capitals, as shown here.

"User" is used to refer to a human user, whereas "client" refers to
the software being run by the user.

In examples, "C:" and "S:" indicate lines sent by the client and
server respectively. If a single "C:" or "S:" label applies to
multiple lines, then the line breaks between those lines are for
editorial clarity only and are not part of the actual protocol
exchange.

As with all IMAP extension documents, the case used in writing IMAP
protocol elements herein is chosen for editorial clarity, and
implementations must pay attention to the numbered rules at the
beginning of [RFC3501] Section 9.

3. FETCH Data Item

3.1. Command

   To retrieve a preview for a message, the "PREVIEW" FETCH attribute is
   used when issuing a FETCH command.

   If no algorithm identifier is provided, the server decides which of
   its built-in algorithms to use to generate the preview text.

   Alternately, the client may explicitly indicate which algorithm(s)
   should be used in a parenthesized list after the PREVIEW attribute
   containing the name of the algorithm. These algorithms MUST be one
   of the algorithms identified as supported in the PREVIEW capability
   responses. If a client requests an algorithm that is unsupported,
   the server MUST return a tagged BAD response.

   The order of the algorithms in the parenthesized list (from left to
   right) defines the client's priority decision. Duplicate algorithms
   in the list SHOULD be ignored. For purposes of duplicate detection,
   priority modifiers (Section 5) should be ignored. A server MUST
   honor a client's algorithm priority decision.

3.2. Response

   The algorithm used by the server to generate the preview is returned
   preceding the preview string.

   The server returns a variable-length string that is the generated
   preview for that message.

   Example: Retrieving preview information in a SELECTed mailbox

      C: A1 FETCH 1 (PREVIEW)
      S: * 1 FETCH (PREVIEW (FUZZY "Preview text!"))
      S: A1 OK FETCH complete.

   A server SHOULD strive to generate the same string for a given
   message for each request. However, since previews are understood to
   be a representation of the message data and not a canonical view of
   its contents, a client MUST NOT assume that a message preview is
   immutable for a given message. This relaxed requirement permits a
   server to offer previews as an option without requiring potentially
   burdensome storage and/or processing requirements to guarantee
   immutability for a use case that does not require this strictness.

   If the preview is not available, the server MUST return NIL as the
   PREVIEW response. A NIL response indicates to the client that

preview information MAY become available in a future PREVIEW FETCH
request. Note that this is semantically different than returning a
zero-length string, which indicates an empty preview.

4. PREVIEW Algorithms

4.1. FUZZY

The FUZZY algorithm directs the server to use any internal algorithm
it desires, subject to the below limitations, to generate a textual
preview for a message.

The FUZZY algorithm MUST be implemented by any server that supports
the PREVIEW extension.

The generated string MUST NOT be content transfer encoded and MUST be
encoded in UTF-8 [RFC3629]. For purposes of this section, a "preview
character" is defined as a single UCS character encoded in UTF-8.

The preview text MUST be treated as text/plain MIME data by the
client.

The server SHOULD limit the length of the preview text to 200 preview
characters. This length should provide sufficient data to generally
support both various languages (and their different average word
lengths) and different client display size requirements.

The server MUST NOT output preview text longer than 256 preview
characters.

The server SHOULD remove any formatting markup that exists in the
original text.

If the FUZZY algorithm generates a preview that is not based on the
body content of the message and the LANGUAGE [RFC5255] extension is
supported by the server, the preview text SHOULD be generated
according to the the language rules that apply to human-readable
text. For example, a message that consists of a single image MIME
part has no human-readable text to generate preview information from.
Instead, the server may wish to output a description that the message
contains an image and describe some attributes of the image, such as
image format, size, and filename. This descriptive text is not a
product of the message body itself but is rather auto-generated data
by the server, and should thus use the rules defined for human-
generated text described in the LANGUAGE extension (if supported on
the server).

5. **PREVIEW Priority Modifiers**

5.1. **LAZY**

   The LAZY modifier directs the server to return the preview
   representation only if that data can be returned without undue delay
   to the client.

   This modifier allows a client to inform the server that preview data
   is nice-to-have, but the server SHOULD NOT block the return of other
   FETCH information at the expense of generating the preview data.

   For example, a client displaying the initial mailbox listing to a
   user may want to display preview information associated with messages
   in that listing. However, this information is secondary to providing
   the mailbox listing, with message details, and the client is willing
   to load any unavailable previews in the background and display them
   as they are provided by the server. In this case, the client would
   apply the LAZY modifier to the desired algorithm(s) to direct the
   server to only return pre-generated preview data so that retrieval of
   the other FETCH information is not blocked by possibly expensive
   preview generation.

   Generally, the LAZY modifier will only be used once per mailbox load
   during the initial listing. If preview information is not available
   during this initial FETCH, the expectation is that a second non-LAZY
   FETCH will take place after mailbox listing activities are complete.
   Thus, a maximum of 2 PREVIEW FETCH queries should occur for any
   message in a selected mailbox. A client SHOULD NOT continually issue
   LAZY PREVIEW FETCH commands in a selected mailbox as the server is
   under no requirement to return preview information for this command,
   which could lead to an unnecessary waste of system and network
   resources. See Example 4 in the Examples section for how this can be
   implemented.

   The LAZY modifier MUST be implemented by any server that supports the
   PREVIEW extension.

6. **Examples**

Example 1: Requesting FETCH without explicit algorithm selection.

```
C: A1 CAPABILITY
S: * CAPABILITY IMAP4rev1 PREVIEW=FUZZY
S: A1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: A2 FETCH 1 (RFC822.SIZE PREVIEW)
S: * 1 FETCH (RFC822.SIZE 20000 PREVIEW (FUZZY {58}
S: This is the first line of text from the first text part.
S: ))
S: A2 OK FETCH complete.
```

Example 2: Requesting FETCH with explicit algorithm selection.

```
C: B1 CAPABILITY
S: * CAPABILITY IMAP4rev1 PREVIEW=FUZZY
S: B1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: B2 FETCH 1 (RFC822.SIZE PREVIEW (FUZZY))
S: * 1 FETCH (RFC822.SIZE 20000 PREVIEW (FUZZY {58}
S: This is the first line of text from the first text part.
S: ))
S: B2 OK FETCH complete.
```

Example 3: Requesting FETCH with invalid explicit algorithm
selection.

```
C: C1 CAPABILITY
S: * CAPABILITY IMAP4rev1 PREVIEW=FUZZY
S: C1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: C2 FETCH 1 (RFC822.SIZE PREVIEW (UNKNOWN-PREVIEW-ALGO))
S: C2 BAD FETCH contains invalid preview algorithm name.
```

Example 4: Use explicit algorithm priority selection, with LAZY
modifier, to obtain previews during initial mailbox listing if
readily available; otherwise, load previews in background.

```
C: D1 CAPABILITY
S: * CAPABILITY IMAP4rev1 PREVIEW=FUZZY
S: D1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: D2 FETCH 1:3 (ENVELOPE PREVIEW (LAZY=FUZZY))
S: * 1 FETCH (ENVELOPE ("Wed, 25 Oct 2017 15:03:11 +0000" [...])
   PREVIEW (FUZZY {58}
S: This is the first line of text from the first text part.
S: ))
S: * 2 FETCH (PREVIEW (FUZZY "") ENVELOPE
   ("Thu, 26 Oct 2017 12:17:23 +0000" [...]))
S: * 3 FETCH (ENVELOPE ("Fri, 27 Oct 2017 22:19:21 +0000" [...])
   PREVIEW (FUZZY NIL))
S: D2 OK FETCH completed.
[...Client knows that message 2 has a preview that is empty;
    therefore, client only needs to request message 3 preview again
    (e.g. in background)...]
C: D3 FETCH 3 (PREVIEW (FUZZY))
S: * 3 FETCH (PREVIEW (FUZZY {27}
S: First sentence of mail 3.
S: ))
S: D3 OK Fetch completed.
```

   Example 5: Retrieve preview information for search results within a
   single mailbox. Use SEARCHRES [RFC5182] extension to save a round-
   trip.

     C: E1 CAPABILITY
     S: * CAPABILITY IMAP4rev1 PREVIEW=FUZZY SEARCHRES
     S: E1 OK Capability command completed.
     [...a mailbox is SELECTed...]
     C: E2 SEARCH RETURN (SAVE) FROM "FOO"
     C: E3 FETCH $ (UID PREVIEW (LAZY=FUZZY))
     S: E2 OK SEARCH completed.
     S: * 5 FETCH (UID 13 PREVIEW (FUZZY {10}
     S: Preview!
     S:))
     S: * 9 FETCH (UID 23 PREVIEW (FUZZY NIL))
     S: E3 OK FETCH completed.
     [...Retrieve message 9 preview in background...]
     C: E4 UID FETCH 23 (PREVIEW (FUZZY))
     S: * 9 FETCH (PREVIEW (FUZZY {18}
     S: Another preview!
     S:))
     S: E4 OK FETCH completed.

7. Formal Syntax

   The following syntax specification uses the augmented Backus-Naur
   Form (BNF) as described in ABNF [RFC5234]. It includes definitions
   from IMAP [RFC3501].

```
   capability         =/ "PREVIEW=FUZZY"

   fetch-att          =/ "PREVIEW" [SP "(" preview-alg-fetch *(SP
                          preview-alg-fetch) ")"]

   msg-att-dynamic    =/ "PREVIEW" SP "(" preview-alg SP nstring ")"

   preview-alg        =  "FUZZY" / preview-alg-ext

   preview-alg-ext    =  preview-atom  ; New algorithm names MUST
                                       ; conform with the
                                       ; recommendations described in
                                       ; RFC 6648, Section 3

   preview-alg-fetch  =  preview-alg / preview-mod "=" preview-alg

   preview-atom       =  1*<any ATOM-CHAR except "=">

   preview-mod        =  "LAZY" / preview-mod-ext

   preview-mod-ext    =  preview-atom  ; New priority modifier names
                                       ; MUST conform with the
                                       ; recommendations described in
                                       ; RFC 6648, Section 3
```

8. Acknowledgements

   The author would like to thank the following people for their
   comments and contributions to this document: Stephan Bosch, Bron
   Gondwana, Teemu Huovila, Steffen Lehmann, Alexey Melnikov, Chris
   Newman, Jeff Sipek, Timo Sirainen, Steffen Templin, and Aki Tuomi.

9. IANA Considerations

   IMAP4 [RFC3501] capabilities are registered by publishing a standards
   track or IESG-approved experimental RFC. The registry is currently
   located at:

      http://www.iana.org/assignments/imap-capabilities

   This document requests that IANA adds the "PREVIEW=FUZZY" capability
   to the IMAP4 [RFC3501] capabilities registry.

   This document requests that IANA create a new PREVIEW algorithms
   registry, which registers preview algorithms by publishing a
   standards track or IESG-approved experimental RFC. This document
   constitutes registration of the FUZZY algorithm in that registry.

This document requests that IANA create a new PREVIEW modifiers
registry, which registers preview modifiers by publishing a standards
track or IESG-approved experimental RFC. This document constitutes
registration of the LAZY modifier in that registry.

10. Security Considerations

Implementation of this extension might enable denial-of-service
attacks against server resources, due to excessive memory or CPU
usage during preview generation or increased storage usage if preview
results are stored on the server after generation. Servers MAY limit
the resources that preview generation uses.

11. References

11.1. Normative References

[RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
            Requirement Levels", BCP 14, RFC 2119,
            DOI 10.17487/RFC2119, March 1997,
            <https://www.rfc-editor.org/info/rfc2119>.

[RFC3501]   Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION
            4rev1", RFC 3501, DOI 10.17487/RFC3501, March 2003,
            <https://www.rfc-editor.org/info/rfc3501>.

[RFC3629]   Yergeau, F., "UTF-8, a transformation format of ISO
            10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November
            2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC5234]   Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax
            Specifications: ABNF", STD 68, RFC 5234,
            DOI 10.17487/RFC5234, January 2008,
            <https://www.rfc-editor.org/info/rfc5234>.

[RFC5255]   Newman, C., Gulbrandsen, A., and A. Melnikov, "Internet
            Message Access Protocol Internationalization", RFC 5255,
            DOI 10.17487/RFC5255, June 2008,
            <https://www.rfc-editor.org/info/rfc5255>.

[RFC6648]   Saint-Andre, P., Crocker, D., and M. Nottingham,
            "Deprecating the "X-" Prefix and Similar Constructs in
            Application Protocols", BCP 178, RFC 6648,
            DOI 10.17487/RFC6648, June 2012,
            <https://www.rfc-editor.org/info/rfc6648>.

   [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
              2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
              May 2017, <https://www.rfc-editor.org/info/rfc8174>.

11.2. Informative References

   [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
              Extensions (MIME) Part One: Format of Internet Message
              Bodies", RFC 2045, DOI 10.17487/RFC2045, November 1996,
              <https://www.rfc-editor.org/info/rfc2045>.

   [RFC2854] Connolly, D. and L. Masinter, "The 'text/html' Media
              Type", RFC 2854, DOI 10.17487/RFC2854, June 2000,
              <https://www.rfc-editor.org/info/rfc2854>.

   [RFC5182] Melnikov, A., "IMAP Extension for Referencing the Last
              SEARCH Result", RFC 5182, DOI 10.17487/RFC5182, March
              2008, <https://www.rfc-editor.org/info/rfc5182>.

Appendix A. Change History (To be removed by RFC Editor before publication)

   Changes from draft-slusarz-imap-fetch-snippet-00:

   o Added standardized language to Section 2 regarding IMAP ABNF
      conventions

   o Changed draft name to draft-ietf-extra-imap-fetch-snippet-##

   Changes from draft-ietf-extra-imap-fetch-snippet-00:

   o Changed nomenclature from "snippet" to "preview"

   o Changed draft name to draft-ietf-extra-imap-fetch-preview-##

   o Update to RFC 8174 boilerplate

   o Updated length requirements for PREVIEW=FUZZY

   o Added preview-atom ABNF to limit use of "=" character

   o UTF-8 is a normative reference

   o Clarify that characters for purpose of length limitations are
      defined as UCS characters as encoded by UTF-8

   o Fix some incorrect literal lengths in examples

Changes from draft-ietf-extra-imap-fetch-preview-00:

o Updated postal address

o Added example to FETCH response section

o Added example on how LANGUAGE extension may influence preview
   generation

o Added recommendation that only one LAZY FETCH be executed for a
   message per mailbox

o Added request to create algorithm and modifier registries

o Added requirement that algorithm and modifier names conform to [RFC
   6648](RFC)

o Added DoS attack info to security considerations

o Distinguish between NIL response and zero-length string

o Don't use deprecated "X-" convention in example

o Spelling and nits

Author's Address

Michael M. Slusarz
Open-Xchange Inc.
530 Lytton Avenue
Palo Alto, California 94301
US

Email: michael.slusarz@open-xchange.com