EXTRA
Internet-Draft
Intended status: Standards Track
Expires: June 12, 2020

M. Slusarz
Open-Xchange Inc.

December 10, 2019

IMAP4 Extension: Message Preview Generation
draft-ietf-extra-imap-fetch-preview-07

Abstract

This document specifies an Internet Message Access Protocol (IMAP) protocol extension allowing a client to request a server-generated abbreviated representation of message data useful as a contextual preview of the entire message.

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 June 12, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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 for the retention period of the source message. 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 indicates support for this extension via the "PREVIEW" capability name.

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 list is provided, the server MUST return data using the text/imap-fetch-preview algorithm.

The client may explicitly indicate which algorithm(s) should be used to generate the preview list via a parenthesized list of algorithm names output after the PREVIEW attribute. Unsupported algorithms in the list MUST be ignored. If the list contains no supported algorithms, the server MUST return a tagged BAD response to the FETCH command.

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.

A server should return preview data for the first algorithm that returns non-empty preview text. Empty preview text is defined as either the NIL response or an empty (zero length) string. If no algorithm produces non-empty preview text, the server should respond with the preview data generated by the final algorithm in the list.

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 (text/imap-fetch-preview "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. For example, the underlying IMAP server may change for an account due to a software upgrade; account state information may be retained in the migration, but the new server may support a different PREVIEW generation algorithm. Thus, message state may remain the same but the FETCH PREVIEW response may change, if that data was not part of the data migration.

It is possible that preview text is not available now, but might be available later -- perhaps the server's preview-generating resources are overloaded, there is a server-imposed timeout during preview generation, or there is some transient issue with fetching the message body. In such cases, the server will return NIL as the preview response, and the client can try to retrieve the preview later.

On the other hand, it is possible that the server has determined that no meaningful preview text can be generated for a particular message, and that decision won't change later. Examples of this involve encrypted messages, content types the server does not support previews of, and other situations where the server is not able to extract information for a preview. In such cases, the server will return a zero-length string. Clients should not send another FETCH for a preview for such messages.

4. PREVIEW Algorithms

4.1. Description

Algorithms MUST be named in media type [RFC6838] format.

This document defines an extension specific algorithm, text/imap-fetch-preview, that outlines precise semantics regarding the generated preview data. Future documents may define additional PREVIEW specific algorithms.

In the absence of a PREVIEW specific algorithm definition, preview generation semantics are server-dependent. If a server supports an algorithm it MUST return preview data in the media type format requested by the client. A client MUST NOT assume that a server supports a preview algorithm not specifically defined in this document.

4.2. text/imap-fetch-preview

The text/imap-fetch-preview 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 text/imap-fetch-preview algorithm MUST be implemented by any server that supports the PREVIEW extension.

The generated preview text MUST be treated as text/plain [RFC2046] media type data by the client.

The generated string MUST NOT be content transfer encoded and MUST be encoded in UTF-8 [RFC3629]. The server SHOULD remove any formatting markup and do what other processing might be useful in rendering the preview as plain text.

For purposes of this section, a "preview character" is defined as a single UCS character encoded in UTF-8. Note: a single preview character may compromise multiple octets, so any buffers implemented to conform to the string limitations identified in this document should be sized to prevent possible overflow errors.

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 diverse client display size requirements.

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

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 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 from which to generate preview information. 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.

The LAZY modifier alters preview generation semantics of the specific algorithm it is applied to. If no algorithm is provided as an argument, the LAZY modifier applies to the text/imap-fetch-preview algorithm.

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
S: A1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: A2 FETCH 1 (RFC822.SIZE PREVIEW)
S: * 1 FETCH (RFC822.SIZE 5647 PREVIEW
   (text/imap-fetch-preview {200}
S: Lorem ipsum dolor sit amet, consectetur adipiscing elit.
S: Curabitur aliquam turpis et ante dictum, et pulvinar dui congue.
S: Maecenas hendrerit, lorem non imperdiet pellentesque, nulla
S: ligula nullam
S: ))
S: A2 OK FETCH complete.
```

Example 2: Requesting FETCH with explicit algorithm selection.

```
C: B1 FETCH 1 (RFC822.SIZE PREVIEW (text/imap-fetch-preview))
S: * 1 FETCH (RFC822.SIZE 91377 PREVIEW
   (text/imap-fetch-preview {53}
S: Preview text generated from message body text data.
S: ))
S: B1 OK FETCH complete.
```

Example 3: Requesting FETCH with unknown explicit algorithm selection.

```
C: C1 CAPABILITY
S: * CAPABILITY IMAP4rev1 PREVIEW
S: C1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: C2 FETCH 1 (RFC822.SIZE PREVIEW (example/unknown))
S: C2 BAD FETCH contains unknown 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 FETCH 1:3 (ENVELOPE PREVIEW (LAZY))
S: * 1 FETCH (ENVELOPE ("Wed, 25 Oct 2017 15:03:11 +0000" [...])
   PREVIEW (text/imap-fetch-preview "Preview text for message 1."))
S: * 2 FETCH (PREVIEW (text/imap-fetch-preview "") ENVELOPE
   ("Thu, 26 Oct 2017 12:17:23 +0000" [...]))
S: * 3 FETCH (ENVELOPE ("Fri, 27 Oct 2017 22:19:21 +0000" [...])
   PREVIEW (text/imap-fetch-preview NIL))
S: D1 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: D2 FETCH 3 (PREVIEW (text/imap-fetch-preview))
S: * 3 FETCH (PREVIEW (text/imap-fetch-preview {30}
S: Message data from message 3.
S: ))
S: D2 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 SEARCHRES
S: E1 OK Capability command completed.
[...a mailbox is SELECTed...]
C: E2 SEARCH RETURN (SAVE) FROM "FOO"
C: E3 FETCH $ (UID PREVIEW (LAZY=text/imap-fetch-preview))
S: E2 OK SEARCH completed.
S: * 5 FETCH (UID 13 PREVIEW (text/imap-fetch-preview "Preview!"))
S: * 9 FETCH (UID 23 PREVIEW (text/imap-fetch-preview NIL))
S: E3 OK FETCH completed.
[...Retrieve message 9 preview in background...]
C: E4 UID FETCH 23 (PREVIEW (text/imap-fetch-preview))
S: * 9 FETCH (UID 23 PREVIEW (text/imap-fetch-preview
   "Another preview!"))
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"

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

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

preview-alg       =  "text/imap-fetch-preview" / preview-alg-ext

preview-alg-ext   =  preview-atom  ; New algorithm names SHOULD be
                                   ; registered with IANA and MUST
                                   ; conform with the media type
                                   ; format described in [RFC6838]

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

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

preview-mod       =  "LAZY"
```

8. 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" capability to the IMAP4 [RFC3501] capabilities registry.

This document requests that IANA create a new "IMAP FETCH PREVIEW Algorithms" registry, which registers preview algorithms by IETF Review [RFC8126]. An assignment consists of the algorithm name (as defined by the preview-alg-ext ABNF entry) and the document that defines the algorithm. This document constitutes registration of the text/imap-fetch-preview algorithm in that registry.

9. 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. In order to mitigate such attacks, servers SHOULD log the client authentication identity on FETCH PREVIEW operations in order to facilitate tracking of abusive clients.

Just as the messages they summarize, preview data may contain sensitive information. If generated preview data is stored on the server, e.g. for caching purposes, these previews MUST be protected with equivalent authorization and confidentiality controls as the source message.

10. References

10.1. Normative References

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[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>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

[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>.

10.2. 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:

- Added standardized language to Section 2 regarding IMAP ABNF conventions
- Changed draft name to draft-ietf-extra-imap-fetch-snippet-##

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

- Changed nomenclature from "snippet" to "preview"
- Changed draft name to draft-ietf-extra-imap-fetch-preview-##
- Update to RFC 8174 boilerplate
- Updated length requirements for PREVIEW=FUZZY
- Added preview-atom ABNF to limit use of "=" character
- UTF-8 is a normative reference
- Clarify that characters for purpose of length limitations are defined as UCS characters as encoded by UTF-8
- Fix some incorrect literal lengths in examples

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

- Updated postal address
- Added example to FETCH response section
- Added example on how LANGUAGE extension may influence preview generation
- Added recommendation that only one LAZY FETCH be executed for a message per mailbox
- Added request to create algorithm and modifier registries
- Added requirement that algorithm and modifier names conform to RFC 6648
- Added DoS attack info to security considerations
- Distinguish between NIL response and zero-length string
- Don't use deprecated "X-" convention in example
- Spelling and nits

Changes from draft-ietf-extra-imap-fetch-preview-01:

- Fix capability ABNF
- Removed CAPABILITY string for examples where it did not add valuable context
- Altered preview data in examples to cover a variety of potential server return scenarios
- Added "SHOULD be registered" language to algorithm names and priority modifiers

Changes from draft-ietf-extra-imap-fetch-preview-02:

- Move Acknowledgments to un-numbered appendix
- Improved abstract text
- Consistently use "priority modifiers" instead of "modifiers"
- Update example to conform with RFC 3501 UID FETCH requirements

Changes from draft-ietf-extra-imap-fetch-preview-03:

- Remove preview modifier registry request
- Improve instructions for registration of algorithms
- Add storage information to security considerations
- Clarify parsing of algorithm list in FETCH command
- Clarify difference between NIL response and zero-length string
- Add normative reference for text/plain
- Add warning regarding buffers and multiple octet preview characters
- Clarify how to handle preview data return when using an explicit algorithm list
- Various editorial fixes

Changes from draft-ietf-extra-imap-fetch-preview-04:

- Make clear that preview caching is tied to retention period of the source message

Changes from draft-ietf-extra-imap-fetch-preview-05:

- Clarify "zero-length string" preview data vs. NIL preview data
- MIME data -> media type
- Capability registration should not include the algorithm name
- Give example of how PREVIEW data might change over time

Changes from draft-ietf-extra-imap-fetch-preview-06:

- Change algorithm names to media types
- FUZZY algorithm changed to text/imap-fetch-preview
- Remove server broadcast of PREVIEW algorithm extensions from capability
- Default, fallback algorithm in absence of client selection now MUST be text/imap-fetch-preview

- LAZY modifier should work on default algorithm if no specific algorithm is provided as an argument

Acknowledgments

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

Author's Address

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

Email: michael.slusarz@open-xchange.com