

Network Working Group
Internet-Draft
Obsoletes: [2088](#) (if approved)
Intended status: Standards Track
Expires: September 7, 2016

A. Melnikov, Ed.
Isode Ltd
March 6, 2016

IMAP4 non-synchronizing literals
draft-ietf-imapapnd-rfc2088bis-04.txt

Abstract

The Internet Message Access Protocol ([RFC 3501](#)) contains the "literal" syntactic construct for communicating strings. When sending a literal from client to server, IMAP requires the client to wait for the server to send a command continuation request between sending the octet count and the string data. This document specifies an alternate form of literal that does not require this network round trip.

This document specifies 2 IMAP extensions: LITERAL+ and LITERAL-. LITERAL+ allows the alternate form of literals in all IMAP commands. LITERAL- is the same as LITERAL+, but disallows the alternate form of literals unless they are 4096 bytes or less.

This document obsoletes [RFC 2088](#).

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 <http://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 September 7, 2016.

Copyright Notice

Copyright (c) 2016 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 (<http://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.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
2.	Requirements Notation	3
3.	Specification	3
4.	Considerations on when to use and not to use synchronizing literals	4
5.	LITERAL- capability	5
6.	Interaction with BINARY extension	6
7.	Interaction with MULTIAPPEND extension	6
8.	Formal Syntax	6
9.	Security Considerations	6
10.	IANA Considerations	7
11.	Acknowledgments	7
12.	References	7
12.1.	Normative References	7
12.2.	Informative References	8
Appendix A.	Changes since RFC 2088	8
	Author's Address	8

1. Introduction

The Internet Message Access Protocol [[RFC3501](#)] contains the "literal" syntactic construct for communicating strings. When sending a literal from client to server, IMAP requires the client to wait for the server to send a command continuation request between sending the octet count and the string data. This document specifies an alternate form of literal that does not require this network round trip.

This document specifies 2 IMAP extensions: LITERAL+ and LITERAL-. LITERAL+ allows the alternate form of literals (called "non-synchronized literals" below) in all IMAP commands. LITERAL- is the same as LITERAL+, but disallows the alternate form of literals unless they are 4096 bytes or less.

2. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [[RFC2119](#)].

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.

3. Specification

The non-synchronizing literal is added as an alternate form of literal, and may appear in communication from client to server instead of the IMAP [[RFC3501](#)] form of literal. The IMAP form of literal, used in communication from client to server, is referred to as a synchronizing literal. The non-synchronizing literal form MUST NOT be sent from server to client.

Non-synchronizing literals may be used with any IMAP server implementation that returns "LITERAL+" or "LITERAL-" as one of the supported capabilities to the CAPABILITY command. If the server does not advertise either of the above capabilities, the client can only use synchronizing literals. The difference between "LITERAL+" and "LITERAL-" extensions is explained in [Section 5](#).

The non-synchronizing literal is distinguished from the original synchronizing literal by having a plus ('+') between the octet count and the closing brace ('}'). The server does not generate a command continuation request in response to a non-synchronizing literal, and

clients are not required to wait before sending the octets of a non-synchronizing literal.

The protocol receiver of an IMAP server MUST check the end of every received line (a sequence of octets that ends with a CRLF) for an open brace ('{') followed by an octet count, a plus ('+'), and a close brace ('}') immediately preceeding the CRLF. If it finds this sequence, it is the octet count of a non-synchronizing literal and the server MUST treat the specified number of following octets and the the following line ([RFC3501](#)) as part of the same command.

It's important to note that the literal is not delimited by CRLF. It ends after the number of bytes specified by the octet count, and the current command continues from there. There might be a CRLF immediately after, which ends the command. Or there might be more octets, specifying other command parameters, before the CRLF. If a SP (space) character is needed between parameters, it's important that the SP appear after the literal, in its appropriate place.

A server MAY still process commands and reject errors on a line-by-line basis, as long as it checks for non-synchronizing literals at the end of each line.

Example:

```
C: A001 LOGIN {11+}
C: FRED FOOTBAR {7+}
C: fat man
S: A001 OK LOGIN completed
```

This is semantically equivalent to this version that uses quoted strings instead of literals:

```
C: A001 LOGIN "FRED FOOTBAR" "fat man"
S: A001 OK LOGIN completed
```

Note that the space after FOOTBAR in the first version corresponds to the space between the two quoted strings in the second.

[4.](#) Considerations on when to use and not to use synchronizing literals

This section is important to understand for both client and server developers of this IMAP extension.

While non-synchronizing literals have clear advantages for clients, such as simplicity of use, they might be more difficult to handle on the server side. When a client uses a non-synchronizing literal that is too big for the server to accept, a compliant LITERAL+ server

implementation has to make a choice between couple non-optimal choices:

1. Read the number of bytes specified in the non-synchronizing literal and reject the command that included the literal anyway. (The server is allowed to send the tagged BAD/NO response before reading the whole non-synchronizing literal.) This is quite wasteful on bandwidth if the literal is large.
2. Send an untagged BYE response explaining the reason for rejecting the literal (possibly accompanied by an ALERT response code in another response) and close the connection. This will force the client to reconnect or report the error to the user. In the latter case the error is unlikely to be understandable to the user. Additionally, some naive clients are known to blindly reconnect in this case and repeat the operation that caused the problem, introducing an infinite loop.

The problem described above is most common when using the APPEND command, because most commands don't need to send lots of data from the client to the server. Some server implementations impose limits on literals (both synchronizing and non-synchronizing) accepted from clients in order to defend against denial-of-service attacks. Implementations can generally impose much lower limits on literal sizes for all commands other than APPEND. In order to address literal size issue in APPEND, this document introduces a new extension "LITERAL-", described in [Section 5](#).

The situation can also be improved by implementing support for the APPENDLIMIT extension [[APPENDLIMIT](#)], which allows a server to advertise its APPEND limit, so that well behaved clients can check it and avoid uploading big messages in the first place.

5. LITERAL- capability

The "LITERAL-" extension is almost identical to "LITERAL+", with one exception: when "LITERAL-" is advertised, non-synchronizing literals used in any command MUST NOT be larger than 4096 bytes. Any literal larger than 4096 bytes MUST be sent as an [RFC 3501](#) synchronizing literal. A "LITERAL-" compliant server that encounters a non-synchronizing literal larger than 4096 bytes proceeds as described in [Section 4](#). If responding to an APPEND command, the tagged BAD response MUST contain the TOOBIG response code [[RFC4469](#)]. If responding with untagged BYE response, it SHOULD include the TOOBIG response code. Note that the form of the non-synchronizing literal does not change: it still uses the "+" in the literal itself, even if the applicable extension is "LITERAL-".

Because "LITERAL-" is a more restricted version of "LITERAL+", IMAP servers MUST NOT advertise both of these capabilities at the same time. (A server implementation can choose to have a configuration option to pick which one to advertise.)

6. Interaction with BINARY extension

[RFC 4466](#) [[RFC4466](#)] updated the non-terminal "literal8" defined in [[RFC3516](#)] to allow for non-synchronizing literals if both [[RFC3516](#)] and "LITERAL+" extensions are supported by the server.

This document also allows use of this extended "literal8" syntax when both [[RFC3516](#)] and "LITERAL-" extensions are supported by the server.

7. Interaction with MULTIAPPEND extension

[RFC 3502](#) [[RFC3502](#)] describes MULTIAPPEND extension and how it can be used with LITERAL+. The LITERAL- extension can be used with the MULTIAPPEND extension in the same way.

8. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur Form (ABNF) notation as specified in [[ABNF](#)].

Non-terminals referenced but not defined below are as defined by [[RFC3501](#)].

```
literal = "{" number ["+"] "}" CRLF *CHAR8
          ; Number represents the number of CHAR8 octets
```

```
CHAR8   = <defined in RFC 3501>
```

```
literal8 = <defined in RFC 4466>
```

9. Security Considerations

Use of non-synchronizing literals can consume extra resources (e.g. memory) on IMAP servers and can be used for denial-of-service attacks. The "LITERAL-" extension partially improved this situation.

This document doesn't raise any other security concerns not already raised by [[RFC3501](#)].

10. 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/imap-capabilities>

This document requests that IANA update the above registry replace the reference for LITERAL+ to point to this document.

This document also requests that IANA add "LITERAL-" capability pointing to this document to the above registry.

11. Acknowledgments

John G. Myers edited the original LITERAL+ extension.

Valuable comments, both in agreement and in dissent, were received from Dave Cridland, Michael M Slusarz, Arnt Gulbrandsen, Jayanthesh S B., Jamie Nicolson, Barry Leiba and SM.

12. References

12.1. Normative References

- [ABNF] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, [RFC 5234](#), January 2008.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [BCP 14](#), [RFC 2119](#), DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", [RFC 3501](#), DOI 10.17487/RFC3501, March 2003, <<http://www.rfc-editor.org/info/rfc3501>>.
- [RFC3516] Nerenberg, L., "IMAP4 Binary Content Extension", [RFC 3516](#), DOI 10.17487/RFC3516, April 2003, <<http://www.rfc-editor.org/info/rfc3516>>.
- [RFC4466] Melnikov, A. and C. Daboo, "Collected Extensions to IMAP4 ABNF", [RFC 4466](#), DOI 10.17487/RFC4466, April 2006, <<http://www.rfc-editor.org/info/rfc4466>>.

[RFC4469] Resnick, P., "Internet Message Access Protocol (IMAP) CATENATE Extension", [RFC 4469](#), DOI 10.17487/RFC4469, April 2006, <<http://www.rfc-editor.org/info/rfc4469>>.

12.2. Informative References

[APPENDLIMIT]

SrimushnamBoovaraghamoorthy, J. and N. Bisht, "The IMAP APPENDLIMIT Extension", [draft-ietf-imapapnd-appendlimit-extension-10](#) (work in progress), January 2016.

[RFC3502] Crispin, M., "Internet Message Access Protocol (IMAP) - MULTIAPPEND Extension", [RFC 3502](#), DOI 10.17487/RFC3502, March 2003, <<http://www.rfc-editor.org/info/rfc3502>>.

Appendix A. Changes since RFC 2088

Added IANA registration.

Updated references. Also updated considerations about interactions of IMAP extensions.

Additional implementation considerations based on the IMAP mailing list discussions.

Added description of a new capability: LITERAL- .

Author's Address

Alexey Melnikov (editor)
Isode Ltd
14 Castle Mews
Hampton, Middlesex TW12 2NP
UK

Email: Alexey.Melnikov@isode.com

