INTERNET-DRAFT
Intended Status: Proposed Standard
Expires: May 26, 2020

Viruthagiri Thirumavalavan
Dombox, Inc.

Nov 26, 2019

Email Address Length
draft-viruthagiri-email-address-length-00

Abstract

This memo formally defines the overall email address maximum length to 254 characters (octets);

Relaxes local-part limitation set by RFC 821/2821/5321;

Obsoletes domain-part limitation set by RFC 821/2821/5321; AND

Uses the mechanism described in RFC 1869 to define an extension with keyword "EAML" to the SMTP service whereby an SMTP server can declare that it is capable of handling longer email addresses.

Status of this Memo

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

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/1id-abstracts.html

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Copyright and License Notice

Table of Contents

1. Definitions

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 [RFC2119].

The term "character" in this document refers to an "octet character".

2 Background

[RFC3696], an informational RFC says:

```
+-----------------------------------------------------------------+
|In addition to restrictions on syntax, there is a length limit on  |
|email addresses. That limit is a maximum of 64 characters (octets) |
|in the "local part" (before the "@") and a maximum of 255          |
|characters (octets) in the domain part (after the "@") for a total |
|length of 320 characters. Systems that handle email should be      |
|prepared to process addresses which are that long, even though they|
|are rarely encountered.                                            |
+-----------------------------------------------------------------+
```

Both [RFC821] and [RFC2821] says:

```
+-----------------------------------------------------------------+
|The maximum total length of a reverse-path or forward-path is 256  |
|characters (including the punctuation and element separators).     |
+-----------------------------------------------------------------+
```

[RFC5321] updated that part with the following text:

```
+-----------------------------------------------------------------+
|The maximum total length of a reverse-path or forward-path is 256  |
|octets (including the punctuation and element separators).         |
+-----------------------------------------------------------------+
```

To comply with [RFC5321] / [RFC2821] / [RFC821], John C. Klensin, the author of RFC3696, filed Errata [1003] with the following text:

```
+-----------------------------------------------------------------+
|In addition to restrictions on syntax, there is a length limit on  |
|email addresses. That limit is a maximum of 64 characters (octets) |
|in the "local part" (before the "@") and a maximum of 255          |
|characters (octets) in the domain part (after the "@") for a total |
|length of 320 characters. However, there is a restriction in       |
|RFC 2821 on the length of an address in MAIL and RCPT commands of  |
|256 characters. Since addresses that do not fit in those fields are|
```

```
|not normally useful, the upper limit on address lengths should     |
|normally be considered to be 256.                                  |
+-------------------------------------------------------------------+
```

There is one issue however. [RFC5321] / [RFC2821] / [RFC821] places the 256 character limit on the path. A path is defined as

Path = "<" [A-d-l ":"] Mailbox ">"

So the forward-path will contain at least a pair of angle brackets in addition to the Mailbox. This limits the Mailbox (i.e. the email address) to 254 characters.

To correct that issue, Dominic Sayers filed Errata [1690] and updated that part with the following text:

```
+-------------------------------------------------------------------+
|In addition to restrictions on syntax, there is a length limit on  |
|email addresses.  That limit is a maximum of 64 characters (octets)|
|in the "local part" (before the "@") and a maximum of 255          |
|characters (octets) in the domain part (after the "@") for a total |
|length of 320 characters. However, there is a restriction in       |
|RFC 2821 on the length of an address in MAIL and RCPT commands of  |
|254 characters. Since addresses that do not fit in those fields are|
|not normally useful, the upper limit on address lengths should     |
|normally be considered to be 254.                                  |
+-------------------------------------------------------------------+
```

3. Maximum Length

The Maximum Length for an email address is

```
+-------------------------------------------------------------------+
|                                                                   |
|                          254 Characters                           |
|                                                                   |
+-------------------------------------------------------------------+
```

Refer Section 2 for more info.

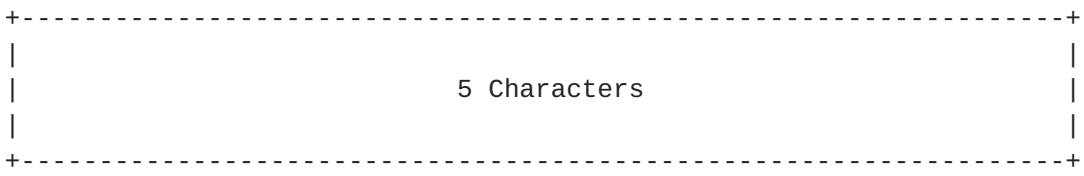
This Maximum Length is a "HARD LIMIT" for end user form inputs. e.g. Websites, Mobile Apps, Gaming Consoles, Smart TVs, Smart Watches, Printers, DSLR cameras etc.

But this is a "SOFT LIMIT" for Receiving Servers. i.e. A receiving server MUST support AT LEAST 254 maximum character email address. A receiving server can extend this limit with the help of EAML SMTP

extension. EAML SMTP extension is defined in Section 7.

4. Minimum Length

The Minimum Length for an email address is



The "Minimum Length" in this section applicable only to DNS-based email addresses. i.e. The sender performs a DNS lookup to fetch MX records before transmitting the mail.

The minimum length in this section applicable only to end user form inputs.

INFORMATIONAL:

The term "email address" is a generic term. It can refer to an Internet address, Intranet address, X.400 address etc.

Internet Address:

An email address on the Internet would look like a@b.c
e.g. john@example.com [user@domain]

Intranet Address:

An email address on the Intranet may look like a@b
e.g. alice@machine50 [user@host]

X.400 Address:

An X.400 address is technically referred to as an Originator/Recipient (OR) address. It consists of several elements, including:

C (Country name)
ADMD (Administration Management Domain, short-form A), usually a public mail service provider
PRMD (Private Management Domain, short-form P)
O (Organization name)
OU (Organizational Unit Names), OU is equivalent to OU0, can have OU1, OU2...
G (Given name)

```
      I     (Initials)
      S     (Surname)

      e.g. G=Harald;S=Alvestrand;O=Uninett;PRMD=Uninett;A=;C=no
```

IP address Literal:

```
      jsmith@[192.168.2.1]
      jsmith@[IPv6:2001:db8::1]
```

5. Obsoleting domain-part limitation

[RFC821] says:

```
+-----------------------------------------------------------------+
|The maximum total length of a domain name or number is 64        |
|characters.                                                      |
+-----------------------------------------------------------------+
```

[RFC2821] updated that part with the following text:

```
+-----------------------------------------------------------------+
|The maximum total length of a domain name or number is 255       |
|characters.                                                      |
+-----------------------------------------------------------------+
```

[RFC5321] updated that part again with the following text:

```
+-----------------------------------------------------------------+
|The maximum total length of a domain name or number is 255       |
|octets.                                                          |
+-----------------------------------------------------------------+
```

Since the maximum total length of an email address is 254 characters, the domain-part limitation of 255 characters is flawed and redundant.

It's flawed because, any such limitation should be less than overall email address length.

i.e. domain-part length < overall email address length

It's redundant because, any email address that has 255 domain characters already an invalid email address due to overall email address length 254 characters.

So this document obsoletes that limitation.

If overall email address length can be defined as "n", then the

maximum total length of a domain name or number is "n-2" characters. We allocated 1 character for the "@" symbol and 1 character for the minimum possible local-part length.

6. Relaxing local-part limitation

[RFC821] says:

```
+-----------------------------------------------------------------+
|The maximum total length of a user name is 64 characters.        |
+-----------------------------------------------------------------+
```

[RFC2821] updated that part with the following text:

```
+-----------------------------------------------------------------+
|The maximum total length of a user name or other local-part is 64  |
|characters.                                                      |
+-----------------------------------------------------------------+
```

[RFC5321] updated that part again with the following text:

```
+-----------------------------------------------------------------+
|The maximum total length of a user name or other local-part is 64  |
|octets.                                                          |
+-----------------------------------------------------------------+
```

This document relaxes the local-part limitation for the reasons found in section 6.1 and 6.2

6.1. Variable Envelope Return Path (VERP)

Most bounce messages have historically been designed to be read by human users, not automatically handled by software. They all convey the same basic idea (the message from X to Y could not be delivered because of reason Z) but with so many variations that it would be nearly impossible to write a program to reliably interpret the meaning of every bounce message. RFC 1894 (obsoleted by RFC 3464) defines a standard format to fix this problem, but support for the standard is far from universal.

[VERP] solves the bounce handling problem.

The hard part of bounce handling is matching up a bounce message with the undeliverable address that caused the bounce. If the mailing list software can see that a bounce resulted from an attempt to send a message to user@example.com then it doesn't need to understand the rest of the information in the bounce. It can simply count how many messages were recently sent to user@example.com, and how many bounces

resulted, and if the proportion of bounced messages is too high, the address is removed from the list.

While bounce message formats in general vary wildly, there is one aspect of a bounce message that is highly predictable: the address to which it will be sent. VERP takes full advantage of this. In a mailing list that uses VERP, a different MAIL FROM address is used for each recipient.

The mailing list manager knows that it sent a message from X (MAIL FROM) to Y (RCPT TO), so if a bounce message is received at address X (MAIL FROM), it can only be because address Y (RCPT TO) was undeliverable, because nothing was sent from X (MAIL FROM) to any other address. Thus the important information has been extracted from the bounce message, without any need to understand its contents, which means the person in charge of the list does not need to deal with it manually.

Typically, an email from Alice to Bob in the above example will have headers like the following:

```
+-----------------------------------------------------------------+
|From: alice@example.org                                          |
|To: bob@example.com                                              |
|Return-Path: bounces@example.org                                 |
+-----------------------------------------------------------------+
```

A very simple VERP address for a mail to bob@example.com will be:

```
+-----------------------------------------------------------------+
|MAIL FROM:<bounces-bob=example.com@example.org>                  |
+-----------------------------------------------------------------+
```

bob is a 3 character local-part, so there won't be any issues.

Let's just assume a user with 62 character local-part subscribe to the list.

A normal mail would look like this.

```
+-----------------------------------------------------------------+
|From: alice@mailchimp.com                                        |
|To: this-electronic-mail-address-local-part-                     |
|          contains-62-characters@example.com                     |
|Return-Path: bounces@example.org                                 |
+-----------------------------------------------------------------+
```

The [VERP] address would look like:

```
+------------------------------------------------------------------+
|MAIL FROM:<bounces-this-electronic-mail-address-local-part-       |
|          contains-62-characters=example.com@example.org>         |
+------------------------------------------------------------------+
```

Above [VERP] address local-part contains 82 characters. If example.com sticks to the RFC limit 64 characters, then the above VERP address structure won't work.

[VERP] was introduced by Daniel J. Bernstein in 1997 and today many software supports VERP.

To name a few,

AmazonSES, CiviCRM, Courier Mail Server, Discourse, exim, ezmlm, GNU Mailman, Inxmail, Mercury Mail Transport System, mlmmj, Mahara, Mailchimp, MediaWiki, Moodle, postfix, qmail, Sendmail, STEdb, StrongMail, Sympa, Thexyz, Zimbra, Target Box, NotifyBC

6.2. Sender Rewriting Scheme (SRS)

Sender Rewriting Scheme ([SRS]) is a scheme for rewriting the envelope sender address of an email message, in view of remailing it. In this context, remailing is a kind of email forwarding. SRS was devised in order to forward email without breaking the Sender Policy Framework (SPF), back in 2003.

[SRS] is a form of variable envelope return path ([VERP]) inasmuch as it encodes the original envelope sender in the local part of the rewritten address. Consider example.com forwarding a message originally destined to bob@example.com to his new address <bob@example.net>

```
+------------------------------------------------------------------+
|ORIGINAL                                                          |
|envelope sender:     alice@example.org                            |
|envelope recipient:  bob@example.com                              |
|                                                                  |
|REWRITTEN                                                         |
|envelope sender:     SRS0=HHH=TT=example.org=alice@example.com    |
|envelope recipient:  bob@example.net                              |
+------------------------------------------------------------------+
```

With respect to VERP, the local part (alice) is moved after her domain name (example.org), further adding a prefix (SRS0), a hash (HHH), and a timestamp (TT)

SRS provides for another prefix, SRS1, to be used for rewriting an already rewritten address, in a multi-hop scenario. If example.net has to forward the message in turn, it can spare adding another timestamp and repeating the original local part (alice). That is, each new forwarder adds just its own hash (HHH) and the domain name of the preceding forwarder:

```
+-----------------------------------------------------------------+
|FURTHER REWRITTEN                                                |
|envelope sender:     SRS1=HHH=example.com==HHH=TT=example.org    |
|                     =alice@example.net                          |
|envelope recipient:  bob@further.example                         |
+-----------------------------------------------------------------+
```

SRS1 incorporates 2 domains in the local-part. So 64 characters are not enough.

6.3. Receiving Servers

[RFC821], Section 4.5.3. says:

```
+-----------------------------------------------------------------+
|There are several objects that have required minimum maximum     |
|sizes.  That is, every implementation must be able to receive    |
|objects of at least these sizes, but must not send objects       |
|larger than these sizes.                                         |
+-----------------------------------------------------------------+

+-----------------------------------------------------------------+
|                                                                 |
|          TO THE MAXIMUM EXTENT POSSIBLE, IMPLEMENTATION         |
|          TECHNIQUES WHICH IMPOSE NO LIMITS ON THE LENGTH        |
|          OF THESE OBJECTS SHOULD BE USED.                       |
|                                                                 |
+-----------------------------------------------------------------+
```

[RFC2821], Section 4.5.3.1 says:

```
+-----------------------------------------------------------------+
|There are several objects that have required minimum/maximum sizes.|
|Every implementation MUST be able to receive objects of at least |
|these sizes. Objects larger than these sizes SHOULD be avoided when|
|possible. However, some Internet mail constructs such as encoded |
|X.400 addresses will often require larger objects: clients MAY   |
|attempt to transmit these, but MUST be prepared for a server to  |
|reject them if they cannot be handled by it. To the maximum extent |
|possible, implementation techniques which impose no limits on the |
|length of these objects should be used.                          |
```

+--+

[RFC5321], Section 4.5.3.1 says:

+--+
|There are several objects that have required minimum/maximum sizes.|
|Every implementation MUST be able to receive objects of at least |
|these sizes. Objects larger than these sizes SHOULD be avoided when|
|possible. However, some Internet mail constructs such as encoded |
|X.400 addresses (RFC 2156) will often require larger objects. |
|Clients MAY attempt to transmit these, but MUST be prepared for a |
|server to reject them if they cannot be handled by it. To the |
|maximum extent possible, implementation techniques that impose no |
|limits on the length of these objects should be used. Extensions |
|to SMTP may involve the use of characters that occupy more |
|than a single octet each. This section therefore specifies lengths|
|in octets where absolute lengths, rather than character counts, are|
|intended. |
+--+

So the 64 character local-part limitation is a "SOFT LIMIT" for receiving servers. i.e. A receiving server MUST support AT LEAST 64 maximum characters in the local-part.

A receiving server SHOULD remove the local-part 64 character limitation whenever possible.

A receiving server can explicitly inform the client that it do not perform the local-part 64 character limitation check with the help of EAML SMTP extension. EAML SMTP extension is defined in Section 7.

6.4. Clients

The primary motivation for relaxing 64 character local-part limitation is to have better bounce handling. Both VERP and SRS are designed to handle bounces.

A bounce message or just "bounce" is an automated message from an email system, informing the sender that the message had not been delivered (or some other delivery problem occurred). The original message is said to have "bounced". More formal terms for bounce message include "Non-Delivery Report" or "Non-Delivery Receipt" (NDR), "Delivery Status Notification" (DSN) message, or a "Non-Delivery Notification" (NDN).

A "bounce address" is also known as MAIL FROM, Return Path, Reverse Path, Envelope From, Envelope Sender etc.

As mentioned in [RFC2821] and [RFC5321], a client MAY attempt to transmit longer email addresses, but MUST be prepared for a server to reject them if they cannot be handled by it.

6.5. Form Inputs

Form Inputs (e.g. Web registration form) rarely check whether the local-part contains 64 characters or not. However, there is a small number of applications certainly do check the local-part character limit.

For example, Embedded Applications (e.g. software found in a DSLR camera), cannot upgrade this limit even if we completely remove that limitation. (We are assuming they already force this 64 character limit)

As mentioned in the last section, the primary motivation for relaxing 64 character local-part limitation is to have better bounce handling. i.e. The MAIL FROM address.

Form inputs however, deals with the RCPT TO address. It is very rare for an end user to have a genuine email address like

this-electronic-mail-address-local-part-contains-62-characters@example.com

For backward-compatibility reasons, this document doesn't mandate removing the local-part limitation for end user form inputs.

However, any kind of end user form inputs SHOULD remove the local-part 64 character limitation WHENEVER POSSIBLE.

7. Framework for the Email Address Maximum Length Extension

This extension is based on the standard SMTP "Service Extensions" described in [RFC1869].

The following service extension is defined:

(1) The name of the SMTP service extension is "Email Address Maximum Length";

(2) The EHLO keyword value associated with this extension is "EAML";

(3) One OPTIONAL parameter is allowed with this EHLO keyword value, a decimal number indicating the maximum email address length in octets that the server will accept.

(4) [RFC822] / [RFC2822] / [RFC5322] header field data is still acceptable to 1000, 998+EOL, where EOL in the SMTP protocol are the control codes, <CR><LF>, carriage return, linefeed. In order to avoid hitting that limit and give some room for header keys, EAML parameter value MUST NOT be greater than 900. i.e. 900 octets is the maximum allowed email address length.

(5) The parameter value SHOULD be from 254-900 (inclusive).

```
S: 220 smtp.example.com ESMTP Ready
C: EHLO bob.example.org
S: 250-smtp.example.com
S: 250-EXPN
S: 250-EAML 500
S: 250-PIPELINING
S: 250 HELP
```

(6) The syntax of the parameter is as follows, using the ABNF notation of [RFC5322] / [RFC2822] / [RFC822]:

```
eaml-param ::= [3DIGIT]
```

(7) If the parameter is omitted or the parameter value is 0-253 (inclusive) or the value is greater than 900, then the value MUST be treated as 254.

(8) Servers offering this extension MUST remove the 64 characters local-part limit. That effectively means, maximum characters allowed in local-part is n-2. Where n is the EAML parameter value.

(9) Servers offering this extension MUST remove the 255 characters domain-part limit. That effectively means, maximum characters allowed in domain-part is n-2. Where n is the EAML parameter value.

(10) The parameter value is OPTIONAL.

```
S: 220 smtp.example.com ESMTP Ready
C: EHLO bob.example.org
S: 250-smtp.example.com
S: 250-EXPN
S: 250-EAML
S: 250-PIPELINING
S: 250 HELP
```

(11) The parameter omitted EAML keyword says:

[*] No local-part limitation.
[*] No domain-part limitation.
[*] 254 maximum characters

(12) No additional SMTP verbs are defined by this extension.

8. IANA Considerations

IANA is hereby requested to register the EAML extension.

9. Security Considerations

This RFC does not discuss security issues and is not believed to raise any security issues not already endemic in electronic mail and present in fully conforming implementations of SMTP.

10. References

10.1. Normative References

[RFC821] Postel, J., "Simple Mail Transfer Protocol", STD 10, RFC 821, DOI 10.17487/RFC0821, August 1982, <http://www.rfc-editor.org/info/rfc821>.

[RFC2821] Klensin, J., Ed., "Simple Mail Transfer Protocol", RFC 2821, DOI 10.17487/RFC2821, April 2001, <http://www.rfc-editor.org/info/rfc2821>.

[RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <http://www.rfc-editor.org/info/rfc5321>.

[RFC822] Crocker, D., "STANDARD FOR THE FORMAT OF ARPA INTERNET TEXT MESSAGES", STD 11, RFC 822, DOI 10.17487/RFC0822, August 1982, <http://www.rfc-editor.org/info/rfc822>.

[RFC2822] Resnick, P., Ed., "Internet Message Format", RFC 2822, DOI 10.17487/RFC2822, April 2001, <http://www.rfc-editor.org/info/rfc2822>.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <http://www.rfc-editor.org/info/rfc5322>.

[RFC1869] Klensin, J., Freed, N., Rose, M., Stefferud, E., and D. Crocker, "SMTP Service Extensions", STD 11, RFC 1869, DOI

10.17487/RFC1869, November 1995, <http://www.rfc-editor.org/info/rfc1869>.

[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>.

10.2. Informative References

[RFC3696] Klensin, J., "Application Techniques for Checking and Transformation of Names", RFC 3696, DOI 10.17487/RFC3696, February 2004, <http://www.rfc-editor.org/info/rfc3696>.

[1003] Klensin, J., "Errata 1003", July 2005, <https://www.rfc-editor.org/errata/eid1003>.

[1690] Sayers, D., "Errata 1690", April 2010, <https://www.rfc-editor.org/errata/eid1690>.

[VERP] Bernstein, D., "Variable Envelope Return Paths", February 1997, <http://cr.yp.to/proto/verp.txt>.

[SRS] Shevek, "The Sender Rewriting Scheme", June 2004, <https://www.libsrs2.org/srs/srs.pdf>.

Authors' Addresses

Viruthagiri Thirumavalavan
Dombox, Inc.

EMail: giri@dombox.org