

Network Working Group  
Internet-Draft  
Intended status: BCP  
Expires: October 27, 2014

M. Kucherawy  
April 25, 2014

Architectural Approaches for Enhancing Email  
draft-kucherawy-email-caps-02

## Abstract

This document provides guidance regarding architectural decisions made when developing enhancements to the Internet message service ("email").

## 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 October 27, 2014.

## Copyright Notice

Copyright (c) 2014 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.

Internet-Draft

Enhancing Email

April 2014

## Table of Contents

<a href="#">1.</a>	Introduction . . . . .	<a href="#">3</a>
<a href="#">2.</a>	Architectural Guidance . . . . .	<a href="#">3</a>
<a href="#">3.</a>	Enhancement History . . . . .	<a href="#">3</a>
<a href="#">4.</a>	The Protocol . . . . .	<a href="#">4</a>
<a href="#">5.</a>	The Message . . . . .	<a href="#">5</a>
<a href="#">6.</a>	Header vs. Envelope . . . . .	<a href="#">5</a>
<a href="#">7.</a>	Deployment Observations and Results . . . . .	<a href="#">7</a>
<a href="#">8.</a>	Consequences of Faulty Design . . . . .	<a href="#">8</a>
<a href="#">9.</a>	Security Considerations . . . . .	<a href="#">9</a>
<a href="#">10.</a>	IANA Considerations . . . . .	<a href="#">9</a>
<a href="#">11.</a>	References . . . . .	<a href="#">9</a>
<a href="#">11.1.</a>	Normative References . . . . .	<a href="#">9</a>
<a href="#">11.2.</a>	Informative References . . . . .	<a href="#">10</a>
<a href="#">Appendix A.</a>	Acknowledgments . . . . .	<a href="#">10</a>

Internet-Draft

Enhancing Email

April 2014

## 1. Introduction

The email service is fully described in [[RFC5598](#)]. It has two core components: The message payload, and the transfer protocol that conveys it.

For various reasons discussed later in this document, it is common for enhancements to the service to be made in undesirable ways. This document first presents some basic architectural recommendations to be considered when enhancing the service, and then describes why these recommendations are apt and provides some history for them.

## 2. Architectural Guidance

When enhancing the email service, it is critical to identify the precise nature of the enhancement. Specifically, an enhancement will affect either the message payload or the way the payload is transferred, but rarely both.

Simply put:

- o An enhancement that affects the content of the message in some way (such as meta-data about how to display the payload, a digital signature ensuring payload integrity, indications of the handling history of the payload, etc.) is best implemented in ways that alter the content somehow, such as addition of header fields or addition of Multipurpose Internet Mail Extensions (MIME) metadata (see [[RFC2045](#)]).
- o An enhancement that purely affects payload transport, and is not meant to be recorded beyond delivery of the message to a mailbox, is best implemented in a way that extend the delivery protocol itself and not in a way that alters the payload.

Enhancements that affect both transport and content are rare, and require special attention to this important boundary.

### [3.](#) Enhancement History

As stated above, the email service is primarily composed of two specifications: The format of the payload, and the method by which the payload is transferred from one handling agent to the next.

The message format was originally fully specified in [[RFC0733](#)], though it has some antecedents in the RFC archive. The current format specification is [[RFC5322](#)]. This format describes two sections: a "header" and a "body". Generally speaking, the body contains the primary content of the message itself, while the header

conveys some metadata such as who sent it, to whom it was sent, where replies are to be directed, how it should be displayed, etc. One notable exception is the Subject: header field, which is essentially part of the content.

The Simple Mail Transfer Protocol (SMTP) was originally fully specified in [[RFC0821](#)], though it too was based on some other previous work. The current specification is [[RFC5321](#)]. The protocol is essentially a simple ASCII dialog between a client system and the server system that exchanges a couple of identifiers -- who the message is "from" and who it is "to" -- and then the message itself, with status codes as the responses at each step.

The partition between these two has often been blurred as a result of the original design and implementation of the service. It was simply not always made clear what the best way is to add extensions.

A number of enhancements to both of these have appeared over the years, which are too numerous to list here. They range in popularity and deployment. Some of these are enhancements to format (such as the addition of multimedia support), others to the protocol (such as enhanced error handling), and a few have augmented both.

The original and increased complexity of the service has led to a body of deployed code that has in turn had some impacts on the development of enhancements over time. This often leads to enhancements that are developed in ways that contradict the advice presented in [Section 2](#). This can have unfortunate consequences, as described below.

#### [4.](#) The Protocol

The Simple Mail Transfer Protocol (SMTP) is the language spoken by email clients and servers to exchange messages. The protocol is all in printable ASCII, which makes it easy for users to "speak" the protocol directly for the purposes of testing, debugging, or illustration.

Essentially, the client introduces itself to the server, which replies with a similar greeting. The client declares that it has a message from a given party for delivery to one or more parties, followed by a declaration that it is ready to send the content. When the server is ready, the client relays its payload (the message) to the server. Finally, the server accepts the message, usually returning a code to the client that uniquely identifies this transaction so that later analysis of the specific transaction is possible. This sequence can repeat if the client has multiple messages to relay during the same SMTP session. When no more

relaying is to be done, the two politely disconnect, and the dialog is complete.

One could make the analogy of a person (perhaps a postal worker) speaking to another person (perhaps at home) and the former handing the latter a sealed envelope bearing a sender address and a recipient address. The contents of the envelope are not known to either of these parties at this stage; the exchange does not require it.

An important point here is that once the exchange is complete, the first party no longer has the message. This is one of the intentional properties of the email service; the message always exists in exactly one place. The notable exception is the period where transmission of the message is complete but not acknowledged; for that brief period, the message exists in two places.

An envelope, in this illustration, can name more than one recipient. An agent holding a message with such an envelope may find it must next relay the message to multiple independent servers to complete delivery to each recipient. In this case, that agent clones ("splits") the envelope, resulting in multiple envelopes each with a subset of the previous recipient set, but with identical content.

## 5. The Message

The email message conveys the content of a message from one or more authors to one or more recipients. The message consists of a header and a body. The body is the primary content, and in modern terms it can contain unstructured plain text, structured multimedia, or nothing at all. The header consists of a set of header fields that include meta-data about the content, such as identifying the party (or parties) that generated it, which agents handled it in transit, the date and time at which it was generated, the (apparent) set of intended recipients, etc. In the case of structured content, the header also contains the initial set of details needed to extract the structure.

If one imagines a printed memo, with fields like "From", "To", "Subject", "Date", and perhaps "Cc", it is easy to envision a simple email message; these fields are at the top, separated by some kind of divider (which might be just an extra blank line or two) followed by the body of the memo. It is in this image that the email format was also created.

## 6. Header vs. Envelope

It is useful to carefully distinguish the separation of function of the message header versus the SMTP envelope when considering the

design of any enhancement to the email service. There are tradeoffs in the choice of enhancement approach. One tends to gain easier adoption, but has less handling control. The other is much more difficult for adoption, but offers much greater handling control.

The most distinctive aspect of the separation is that the addresses in the envelope, used during transfer, can be entirely different from the addresses contained in the message header. So the SMTP return address (MAIL FROM) can be different from the message author (From: header field), and the list of SMTP recipient addresses (RCPT TO) can be entirely different from the recipients listed in the message header (To, Cc, and Bcc header fields).

Thus, what's in that example printed memo in the previous section is completely independent of what was on the envelope that contained it.

The memo might say "From: Alice" and "To: Bob", while the envelope said "From: Charlie" and "To: Deborah". More generally, there is no guarantee that the content and the transport have any relationship at all.

An example of non-core material that is rightly a property of the message and not the envelope includes digital signatures of the payload. One might think of the mark or seal of a notary, which is meant to certify the content and not the envelope containing it.

SMTP also has the notion of "Trace Information" which is a record of the agents that handled the message prior to delivery and when they each processed the message. One might think of a premium package handling service that includes tracking as part of its product, showing through which stations the package was carried and a date/time at each. Email trace information fulfills the same goal, and is normally recorded as Received header fields.

Also recorded in the header, at the time of delivery only, is the "from" portion of the envelope, to permit a reply to be sent to the correct place. This is recorded in a field called Return-Path.

Any message can be forwarded by a user or a piece of software (such as a mailing list service). In this case, it is appropriate to think of the message as taking on a new life beyond its original delivery; that is, it is delivered to the entity that will forward it, and takes on a new life, with a new envelope and possibly a new or revised header, or even augmented content. Caution must be taken when constructing a new header so that information relevant only to the original delivery does not get forwarded; this leakage of information can lead to mishandling of the content or even leakage of private information to the new recipient(s). [[RRVS](#)] provides an example of such risks.

## [7.](#) Deployment Observations and Results

As the email service grew in popularity, it also became a popular target for abuse. In particular, it became a vector for delivery of unwanted commercial email ("spam") or even malicious active content ("malware", such as viruses or worms). These attempt to exploit user trust (and naivete) in order to deliver undesirable content. Among other things, false or misleading From and Subject fields on messages

are commonplace.

Mail User Agents (MUAs) retrieve messages from message stores, and not from the Message Transfer Agents (MTAs) or Message Delivery Agents (MDAs) that affect transport and delivery of messages. They do not have access to the parameters exchanged during the protocol sessions that resulted in the delivery. This led to various enhancements done as message header fields, rather than enhancements to SMTP, or to MUA access protocols such as the Internet Message Access Protocol (IMAP) or Post Office Protocol (POP).

The rise in abusive emails, with the abuse almost entirely aimed at exploiting deficiencies in content handling and presentation (see [\[RFC7103\]](#)), produced a requirement for email-handling agents (primarily MTAs) to be enhanced with powerful mechanisms for analyzing and even modifying messages. Given the considerable range of different ad hoc enhancements that have been made to message formats, discussed above, this requires significant flexibility in the mechanisms for making decisions about, or even altering, header fields in messages as they are processed. By contrast, very little in the way of messaging abuse takes place via misuse of SMTP or its extensions.

Furthermore, SMTP is the infrastructure mechanism for message handling, and infrastructures are always markedly more difficult to modify, especially when the infrastructure is under a series of independent administrative controls, but must somehow come to be coordinated in their enhancements. This again contrasts with the handling of the payload itself, where only the agent generating the content and the agent that will ultimately interpret it -- by presenting it to a user -- need to understand it.

This has resulted in the current environment, in which it is often very easy to add, alter, remove, and analyze header fields on a message, and typically very difficult if not impossible to add or process an SMTP extension for which built-in support does not already exist.

An MTA or MDA advertises the SMTP extensions it supports, through the EHLO command reply. A client that supports a particular extension

can therefore easily determine its applicability with the server with



which it is interacting. If that agent does not include such support, the current agent must decide to do one of two things:

- a. consider the delivery a failure, and begin processing it as an error; or
- b. relay the message anyway, losing the capability afforded by the extension.

In contrast to this negotiation mechanism at the level of SMTP, there is no control exchange for support of header field enhancements. They are present or not, and the client agent has no way to determine whether its semantics are supported by the next handling agent (or the recipient). However an MTA or MDA that does not understand a particular header field will almost always simply ignore that header field and continue to relay it, usually unmodified, to software downstream that does recognize the field and how to use its contents. A good example of this is MIME, whose header fields are typically of use only to MUAs and are ignored by MTAs and MDAs. Moreover, MUAs typically do not include header fields they don't recognize in the material ultimately presented to the end user.

Enhancements done using header fields can be enormously useful when one wishes to deploy a new capability that will not affect or be affected by non-participating agents and is not intended for direct human consumption.

As a result, it is common to assume that adding a new capability to the email service is best accomplished by creating (and hopefully, registering) a new header field specific to that purpose, even if that capability would more properly be implemented as an SMTP extension.

In a few very rare cases, new capabilities have even been developed that include both header field and SMTP extension forms. [[RRVS](#)] again serves as a useful example.

## 8. Consequences of Faulty Design

Using the header for enhancements that do not fit the envelope vs. content model may be convenient given the current deployed environment, but they result in such issues as:

- o inadvertent leakage of data not relevant to later message recipients if the message gets forwarded;

- o no guarantee that any agent in the handling path understands the enhancement or the details associated with it, leading to unexpected results;
- o for messages going to multiple recipients, the possible inadvertent revelation of private information when the message is "fanned out".

For more discussion, see [Section 7.2 of \[RFC5321\]](#), [Section 3.6.3 of \[RFC5322\]](#), and Section 7 of [\[RRVS\]](#).

MTA and MDA implementers need to ensure that SMTP extensions can be added and handled via the runtime environment as easily as they can be for header fields. This will ensure the more sound architectural decisions can be made by designers and operators of future enhancements.

## [9.](#) Security Considerations

An important observation is that the envelope and the header overlap in only a small number of key ways:

- o The Return-Path header field, added at time of delivery, which includes the sender address as extracted from the message envelope; and
- o The Received header field, which might contain the envelope recipient for messages addressed to a single mailbox.

Typically, all other envelope details are discarded upon delivery. Because of this, data about transport that should be ephemeral but are stored in header fields can fall into the wrong hands when the message is forwarded. Following the recommendations above can help to reduce this concern.

## [10.](#) IANA Considerations

This document contains no actions for IANA.

[RFC Editor: Please remove this section prior to publication.]

## [11.](#) References

### [11.1.](#) Normative References

[RFC5598] Crocker, D., "Internet Mail Architecture", [RFC 5598](#),

July 2009.

Kucherawy

Expires October 27, 2014

[Page 9]

Internet-Draft

Enhancing Email

April 2014

## [11.2.](#) Informative References

- [RFC0733] Crocker, D., Vittal, J., Pogran, K., and D. Henderson, "Standard for the format of ARPA network text messages", [RFC 733](#), November 1977.
- [RFC0821] Postel, J., "Simple Mail Transfer Protocol", STD 10, [RFC 821](#), August 1982.
- [RFC2045] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", [RFC 2045](#), November 1996.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", [RFC 5321](#), October 2008.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", [RFC 5322](#), October 2008.
- [RFC7103] Kucherawy, M., Shapiro, G., and N. Freed, "Advice for Safe Handling of Malformed Messages", [RFC 7103](#), January 2014.
- [RRVS] Mills, W. and M. Kucherawy, "The Require-Recipient-Valid-Since Header Field and SMTP Service Extension", [draft-ietf-appsawg-rrvs-header-field](#) (work in progress), April 2014.

## [Appendix A.](#) Acknowledgments

Dave Crocker and John Levine provided useful review comments during the development of this work.

### Author's Address

Murray S. Kucherawy  
270 Upland Drive  
San Francisco, CA 94127  
USA

EMail: superuser@gmail.com

KucheraWy

Expires October 27, 2014

[Page 10]