

Message ORGanization Working Group
Internet-Draft
Updates: [3464](#) (if approved)
Intended status: Standards Track
Expires: January 7, 2010

B. Leiba
Huawei Technologies
July 6, 2009

SMTP Service Extension for Message Recall
draft-leiba-morg-message-recall-00

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/ietf/lid-abstracts.txt>.

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

This Internet-Draft will expire on January 7, 2010.

Copyright Notice

Copyright (c) 2009 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 in effect on the date of publication of this document (<http://trustee.ietf.org/license-info>). Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Abstract

End users occasionally send email messages that they later want to recall, perhaps because they sent incorrect information, or had

Internet-Draft SMTP Service Extension for Message Recall

July 2009

second thoughts about what they said in the message. Proprietary email systems often provide such a recall function. This document specifies a standard mechanism for providing it with Internet email.

Note

A revised version of this draft document will be submitted to the RFC editor as a Proposed Standard for the Internet Community. Discussion and suggestions for improvement are requested, and should be sent to morg@ietf.org.

Table of Contents

1.	Introduction and Informative Discussion	4
2.	Conventions used in this document	6
3.	SMTP service extension keyword	6
4.	The Message-Verification header field	6
5.	The SMTP RECL command	7
5.1.	RECL HOLD	8
5.2.	RECL RELEASE	8
5.3.	RECL RECALL	9
6.	Delivery Status Notifications (DSNs)	10
7.	Timing issues with holding messages	11
8.	Examples	11
8.1.	Example of a Message-Verification header field	11
8.2.	Example of a SMTP RECL transaction	12
8.3.	Example of a message recall DSN	12
9.	Formal Syntax	13
10.	Security Considerations	14
11.	IANA Considerations	15
11.1.	SMTP service extension registration	15
11.2.	Message header field registration	15

11.3.	DSN actions registry creation	15
11.4.	DSN actions value registrations	16
12.	Acknowledgements	16

13.	Normative References	16
	Author's Address	17

1. Introduction and Informative Discussion

End users occasionally send email messages that they later want to recall, perhaps because they sent incorrect information, or had second thoughts about what they said in the message. Proprietary email systems often provide such a recall function, but providing it on the Internet, with a diversity of software, administrative domains, and policies, presents a different set of difficulties. This section will describe some of those difficulties, to help in understanding the reasoning behind the protocol that follows.

By "recall", here, we mean that the sender wants to withdraw the message from the inbox of one or more recipients, such that those recipients no longer have access to the message, nor knowledge of its existence. We assume that a message already seen by a recipient can not be recalled from that recipient.

A system that resides within one administrative domain can easily do record keeping and authentication that will allow it to identify messages and users, and to determine authorization to recall a particular message. It is much harder to do that on the Internet, when the sender is not known to the receiving domain, and when the sending and receiving domains do not have a mutual trust relationship. A protocol that allows the sender to act on messages that have already arrived at the receiving domain must satisfactorily resolve the question of authorization to act on those messages.

Proprietary systems often request message recall by sending a special message. Doing it in that way on the Internet is problematic,

because there's no way to determine in advance that the receiving server supports the protocol and will understand the special message. If it does not, the message will likely be delivered to the actual recipient in addition to the original message that was meant to be recalled.

Further, a new end-to-end protocol to perform this function is not practical. Because Internet email [[SMTP](#)] uses a store-and-forward mechanism to transfer email messages, it might not be possible for the sender to contact the receiving server directly. Indeed, it's often the case that it's not even possible for the sender to identify the receiving server at all.

By defining this protocol as an extension to SMTP, we fit on top of the existing store-and-forward mechanism, and we detect when a server in the store-and-forward chain does not support this function. We then report the status back to the requester asynchronously, using delivery status notifications [[DSN](#)].

A single email message can be sent to a large number of recipients, possibly all at different domains. Some of the recipients might have already seen the message at the time it's being recalled. These can make the process of recalling the message rather difficult. While it's certainly an easy matter to send a recall request to all the receiving domains and have the message recalled from only some of them, depending upon support for this extension and "seen" status of the message by each recipient, this might not be what the sender wants.

It's easy to come up with scenarios where it would be very bad to recall it from some, but not from all recipients -- perhaps doing so would cause confusion, or even embarrassment. It's also easy to come up with scenarios where it would be awkward or embarrassing for anyone to know you were trying to recall the message, in a case where the request is ultimately unsuccessful.

To solve that, we define a two-step process, first putting a "hold" on the message, and then either rescinding that hold (if we don't get the desired result) or proceeding with the recall. The "hold" step is optional, and will not always be needed. Of course, any asynchronous multi-step process must have a mechanism for resolving

hanging requests and missing responses. The protocol suggests using timeouts for those.

Expectations have to be kept reasonable, given the likelihood that most recalls will not be fully successful, and especially not so at first. Initially, most systems won't support this. Even when more do, the success will be lower the more recipients a message has, and the longer the time between send and recall. The limitations will have to be explained to the users as part of the interface they get when they press the "recall this message" button (or whatever the UI mechanism is). If the user is given the choice of "recall what you can" (that is, skip the optional "hold" step) and "all or nothing" (do the hold, and only recall if all holds work), that choice must be simply put and easy to select. Most users will have a hard time understanding the finer points here.

It's probably also a good idea to give the user a summary, probably in the form of an email message, when the recall process is done: a message saying something like, "Of the 10 recipients, the messages appears to have been recalled from 5. 2 appear to have already seen the message, and 3 of the mail systems do not support the recall option." The message might also include the response given for each recipient, in detail. By wording things in terms of "appear to", or the like, and avoiding absolute "was recalled" statements, we can avoid troubles in the case of improper implementations -- perhaps intentionally so (see [Section 10](#)).

Finally, the Security Considerations section describes issues that need to be considered when implementing and deploying this protocol.

[2.](#) Conventions used in this document

In examples, "C:" indicates lines sent by a client that is connected to a server. "S:" indicates lines sent by the server to the client. In this protocol, both client and server will often be Message Transfer Agents (MTAs), but one will be acting as the client and one as the server for the transaction.

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 [RFC 2119](#) [[Kwds](#)].

[3.](#) SMTP service extension keyword

An SMTP server that supports this extension includes the extension keyword "RECL" in its EHLO response.

[4.](#) The Message-Verification header field

If the sending side supports this protocol and wants a message to be recallable, it MUST include a Message-ID header field [[Format](#)] in the message. It MUST also include a Message-Verification header field. The Message-Verification header field contains the following required tags, separated by semicolons (with no white space within):

hash=[string] -- the algorithm used to produce the hash value in the "guid" tag. Current valid values are "SHA1" and "SHA256" [[SHA](#)].

guid=[base64] -- the base64 encoding [[Base64](#)] of the hash of a globally unique ID (GUID) for the message, using the hash algorithm specified in the "hash" tag. The actual GUID (the pre-image of the hash) is kept secret by the sending side.

See [Section 9](#) for the formal syntax of the Message-Verification field.

The Message-Verification header field may be put there by the sender or by the sender's domain. The unhashed GUID MUST be retained and associated with the message, and available to the recalling entity, in order for a recall request to be sent later. To simplify

bookkeeping, the GUID MAY be discarded at some time, after which the message can no longer be recalled.

Recall requests will be accompanied by the target message's Message-ID header field value and the unhashed GUID of the message. The receiving server will locate the message using the Message-ID, retrieve the message's Message-Verification header field, hash the GUID using the hash algorithm specified in that header field, and

compare the new hashed value with the hashed value in that header field. The receiving server considers the message to have been identified ONLY if these tests match. If they do not match, the receiving server MUST consider the message as not found.

[5.](#) The SMTP RECL command

When a message is to be recalled, the recalling entity obtains the Message-ID and (unhashed) GUID of the message. If the recalling entity is not the requester (the end user, in most cases), it is the responsibility of the recalling entity to determine the requester's authorization to recall the message.

The recalling entity then initiates a recall request using SMTP [[SMTP](#)]. The recall request is propagated through SMTP's store-and-forward mechanism, just as a normal SMTP mail transfer is.

A recall request starts with EHLO, MAIL FROM, and RCPT TO, as for the original message. The recall session MUST use EHLO, and the extension list returned MUST be checked to determine that this protocol is supported by the downstream server. If it is not, and the recall request verb is not RELEASE, the current server MUST send back a "BAD" DSN (see [Section 6](#)). DSNs MUST NOT be returned for RELEASE requests.

In place of the DATA command, the recall request uses the new RECL command, as one of the following:

RECL HOLD Message-ID GUID

RECL RELEASE Message-ID GUID

RECL RECALL INFORM [NO | FAILURE | SUCCESS | ALL] Message-ID GUID

"Message-ID", above, is the value of the Message-ID field of the original message, including the angle-brackets. "GUID" is the unhashed GUID of the original message. See [Section 9](#) for the formal syntax of the RECL command.

The RECL command ends the recall request, and is followed by a QUIT

command, to terminate the SMTP session, or an RSET command, to start a new SMTP transaction.

[5.1.](#) RECL HOLD

RECL HOLD requests a hold on the message. For each recipient, a DSN is sent back with a status of "OK", "NO", or "BAD" (see [Section 6](#)). Note that a receiving server might choose not to implement RECL HOLD (see item 1, below).

When a RECL HOLD arrives at the receiving server for a particular recipient, the receiving server MUST do one of the following:

1. Respond with a "NO" DSN because it does not implement RECL HOLD.
2. Determine that the target message is not found, and respond with a "NO" DSN.
3. Determine that the message is already on hold, and respond with an "OK" DSN. Timeouts associated with the held message (see [Section 7](#)) SHOULD be reset the first time this happens.
4. Determine that local policy forbids a hold on the target message, and respond with a "NO" DSN. If policy permits a hold, it MUST also permit a subsequent recall.
5. Determine that the message is already marked as seen, and respond with a "NO" DSN.
6. Determine that for some other reason, the message can not be held, and respond with a "NO" DSN.
7. If none of the above apply, hold the message and respond with an "OK" DSN. Holding the message means putting it in a state where neither the contents of the message nor its presence can be seen by the user, but where it can be restored to its original state when needed.

By holding the message and responding with an "OK" DSN, the receiving server MUST guarantee that the message will be successfully recalled if a subsequent RECL RECALL is received in a timely manner.

[5.2.](#) RECL RELEASE

RECL RELEASE is used to rescind a HOLD, perhaps because they didn't all come back as OK.

When a RECL RELEASE arrives at the receiving server for a particular recipient, the receiving server MUST do one of the following:

1. Determine that the target message is on hold, and restore it to its original, state.
2. Otherwise, ignore the request.

No DSNs are returned for RECL RELEASE.

[5.3.](#) RECL RECALL

RECL RECALL requests that the message be recalled. The recall may fail because of policy, because the message has already been seen by the recipient, and perhaps for other reasons. The recipient may have been notified of the message, but not actually seen the message yet (perhaps through Sieve notifications, or some other mechanism). Whether such a message is recallable is a matter of policy.

The value of the INFORM option works thus:

- NO -- Never inform the recipient that a recall was requested.
- FAILURE -- Inform the recipient that a recall was requested only if the recall fails. This can advise the recipient that the sender wants to withdraw the message, in the event that it could not actually be withdrawn.
- SUCCESS -- Inform the recipient that a recall was requested only if the recall succeeds. This can have the effect of replacing the recalled message with a notification that the message was withdrawn.
- ALL -- Always inform the recipient that a recall was requested, whether it succeeded or not.

The mechanism used to inform the recipient is up to the implementation. It MAY use an email message, but MAY use other locally available notification mechanisms.

When a RECL RECALL arrives at the receiving server for a particular recipient, the receiving server MUST do one of the following:

1. Determine that the target message is not found, and respond with a "NO" DSN.

2. Determine that the message is already on hold (see above), delete it, and respond with an "OK" DSN.

3. Determine that local policy forbids recalling this message, and respond with a "NO" DSN.
4. Determine that the message is already marked as seen, and respond with a "NO" DSN.
5. Determine that for some other reason, the message can not be recalled, and respond with a "NO" DSN.
6. If none of the above apply, delete the message and respond with an "OK" DSN.

If the server responds with an "OK" DSN, it MUST remove the message from the recipient's mailbox, and never put it back. However, no guarantee is made for any other recipients of the message, and it is, of course, possible that one of them might re-send the message to this recipient. Such action would be independent of the recall process.

6. Delivery Status Notifications (DSNs)

Because the recall protocol is fully asynchronous, all responses come in the form of Delivery Status Notifications [[DSN](#)], which are sent back to the MAIL FROM address in the recall request. These DSNs contain action and status codes, as described below, which inform the requesting entity of the success status of the request. When constructing the DSN multipart/report [[Report](#)], only sections [1](#) (human-readable message) and 2 (machine parsable report) are used. The optional [section 3](#) (returned message) MUST NOT be included.

The action code is placed in the "Action" field of [section 2](#), and comprises the recall verb ("HOLD" or "RECALL"; DSNs are not sent for RELEASE commands), one space character (ASCII 0x20), and one of the following:

OK -- The request was successful. The message has been held or recalled.

- NO -- The request was not successful. The DSN MAY include human-readable text in [section 1](#) that further explains the problem.
- BAD -- A server in the SMTP store-and-forward path does not support the RECL protocol.

See [Section 9](#) for the formal syntax of the Action field.

Leiba

Expires January 7, 2010

[Page 10]

Internet-Draft SMTP Service Extension for Message Recall

July 2009

The status code [[Codes](#)] is placed in the "Status" field of [section 2](#), and relates to the action code as follows:

2.0.0 -- for "OK" actions.

5.0.0 -- for "NO" actions.

5.3.3 -- for "BAD" actions.

[7.](#) Timing issues with holding messages

[[anchor9: Should we suggest "reasonable" time values here? If so, what? One suggestion is that we suggest defaults, and include a mechanism in the protocol to specify times -- a time-stamp on the HOLD request, asking for a hold until that time; a time-stamp on the response, saying "I'll hold at least until this time." There'd have to be a tolerance for out-of-sync clocks... in other words, the times would be approximate, not held to the second.]]

If the requesting entity uses RECL HOLD, it MUST keep track of the asynchronous DSNs and decide in a timely manner to RELEASE or RECALL the message. This means having a notion of a timeout period for missing DSNs. A requesting entity MAY selectively re-issue a RECL HOLD once, to those recipients for which no DSN is received within a reasonable time, but it MUST NOT continue doing so repeatedly.

Similarly, an OK response to a HOLD gives the recipient side the responsibility of determining how long to wait for further action. The message SHOULD [[anchor10: MUST?]] automatically be released from hold after a reasonable time if no RELEASE or RECALL is received.

[8.](#) Examples

[8.1.](#) Example of a Message-Verification header field

This shows an example of partial message headers, with a Message-ID field and a Message-Verification field. The message's unhashed GUID is "G9Kw8iJ37Q1027msa4NbU".

```
From: alice@example.org
To: bob@example.com
Subject: Save the date!
Message-ID: <411699893-1246577932-871827273@example.org>
Message-Verification: hash=SHA1;guid=BAv9A56z4M0FU3T/Qn+dw7ck9bA=
```

Leiba

Expires January 7, 2010

[Page 11]

Internet-Draft SMTP Service Extension for Message Recall

July 2009

[8.2.](#) Example of a SMTP RECL transaction

This will request a recall of a message with the above headers. Note that we use the Message-ID value as specified in the original header, and the unhashed GUID. Also note that the final "250 2.0.0 OK" does NOT mean that the message has been recalled, but only that the recall REQUEST was transmitted successfully. A DSN will be sent back to alice@example.org with the actual recall status.

```
C: [connects to mail.example.com]
S: 220 mail.example.com -- ESMTP (Messaging Server version 1.7a)
C: EHLO example.org
S: 250-mail.example.com
S: 250-PIPELINING
S: 250-RECL
S: 250-DSN
S: 250 ENHANCEDSTATUSCODES
C: MAIL FROM:<alice@example.org>
S: 250 2.5.0 OK
C: RCPT TO:<bob@example.com>
S: 250 2.1.5 OK
C: RECL RECALL <411699893-1246577932-871827273@example.org>
  G9Kw8iJ37Q1027msa4NbU
S: 250 2.0.0 OK
C: QUIT
```

S: 221 2.3.0 Goodbye.

[8.3.](#) Example of a message recall DSN

This might be a response to the example recall request above.

Leiba

Expires January 7, 2010

[Page 12]

Internet-Draft SMTP Service Extension for Message Recall

July 2009

To: alice@example.org
From: postmaster@mail.example.com
Subject: Recall Notification (RECALL OK) for bob@example.com
Content-Type: multipart/report; report-type=delivery-status;
boundary=abcde
MIME-Version: 1.0

--abcde

Content-type: text/plain; charset=us-ascii

Your message (guid G9Kw8iJ37Q1027msa4NbU)
was successfully recalled from bob@example.com.

--abcde

Content-type: message/delivery-status

Reporting-MTA: dns; mail.example.com
Original-Envelope-ID: G9Kw8iJ37Q1027msa4NbU

Original-Recipient: [rfc822](#);bob@example.com
Final-Recipient: [rfc822](#);bob@example.com
Action: RECALL OK
Status: 2.0.0

--abcde--

[9.](#) Formal Syntax

The following syntax specification uses the augmented Backus-Naur Form (BNF) as described in [\[ABNF\]](#). Terms not defined here are taken from the Simple Mail Transfer Protocol [\[SMTP\]](#) and Internet Message Format [\[Format\]](#) specifications.

```
action-hold    = "HOLD" SP ("OK" / "NO" / "BAD")
action-recall  = "RECALL" SP ("OK" / "NO" / "BAD")
action-value /= action-hold / action-recall
               ; extends "action-value" from RFC 3464 \[DSN\]
base64string   = 1*(ALPHA / DIGIT / "+" / "/" ) [ "=" [ "=" ] ] CRLF
guid           = dot-atom-text
```

```
guid-hash      = "guid=" base64string
hash-alg       = "hash=" ("SHA1" / "SHA256")
message-verification = "Message-Verification:" [CFWS] hash-alg ";"
                  guid-hash [CFWS] CRLF
recl           = "RECL" SP recl-verb SP message-id SP guid
recl-hold      = "HOLD"
```

```
recl-recall    = "RECALL" SP "INFORM" SP ("NO" / "FAIL" / "ALL")
recl-release   = "RELEASE"
recl-verb      = recl-hold / recl-release / recl-recall
```

[10.](#) Security Considerations

It's possible for an intermediate MTA to add or replace a Message-Verification header field during the transmission of an original message, which could later allow an unauthorized party to recall the message. Since intermediate MTAs can do plenty of other nasty things, including not relaying the message in the first place (but saying they did), this is probably not a real issue.

If the sending domain (as opposed to the sending user agent) creates the Message-Verification header field, it will also be the sending domain that will have to supply the GUID on the RECL command. There will then be no in-band authentication of the user requesting the recall. In this case, it's the responsibility of the sending domain to ensure that the user is authorized to do this.

Timeouts are required, as noted above (see [Section 7](#)).

It's possible for the receiving end to ignore the INFORM option, and to inform the recipient when the sender would not want that. It's possible for the receiving end to send an "OK" DSN but fail to recall the message (perhaps also informing the recipient of this). While these behaviours are contrary to this specification, they are a fact of life when asking for such things on the Wild, Wild Internet. Those who would try to recall messages quietly should bear this in mind.

[11.](#) IANA Considerations

[11.1.](#) SMTP service extension registration

This document defines an SMTP service extension, and IANA is asked to add an entry to the SMTP Service Extensions registry, as follows:

Keyword: RECL
Description: Message recall
Reference: [[IANA: Insert this RFC number.]]
Parameters: (none)

[11.2.](#) Message header field registration

This document defines a new message header field, and IANA is asked to add an entry to the Permanent Message Header Field Names registry, as follows:

Header field name: Message-Verification
Protocol: mail
Status: standard
Reference: [[IANA: Insert this RFC number.]]

[11.3.](#) DSN actions registry creation

This document defines a new DSN action field value, and there is currently no registry for those values. IANA is asked to create a "DSN Actions" sub-registry in the Mail Parameters registry (<http://www.iana.org/assignments/mail-parameters>). New values will be registered using the "Specification Required" policy [[IANA](#)]. The template to be used to register new values is as follows:

To: iana@iana.org
Subject: Registration of new DSN action field value
Value: [the text value of the field]
Reference: [identifies the specification that defines this value]

IANA is asked to enter the following initial values in this registry, all taken from [RFC 3464](#):

Value: failed
Reference: [RFC 3464](#)

Value: delayed
Reference: [RFC 3464](#)

Value: delivered
Reference: [RFC 3464](#)

Value: relayed
Reference: [RFC 3464](#)

Value: expanded
Reference: [RFC 3464](#)

11.4. DSN actions value registrations

This document defines new DSN action field values, and IANA is asked to add the following entries to the DSN Actions registry, created above:

Value: hold [ok/no/bad]
Reference: [[IANA: Insert this RFC number.]]

Value: recall [ok/no/bad]
Reference: [[IANA: Insert this RFC number.]]

12. Acknowledgements

The author gratefully acknowledges feedback provided by Ned Freed and Nathaniel Borenstein on the initial version of this specification.

13. Normative References

- [ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", [RFC 5234](#), January 2008.
- [Base64] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", [RFC 4648](#), October 2006.
- [Codes] Vaudreuil, G., "Enhanced Mail System Status Codes", [RFC 3463](#), January 2003.
- [DSN] Moore, K. and G. Vaudreuil, "An Extensible Message Format for Delivery Status Notifications", [RFC 3464](#), January 2003.
- [Format] Resnick, P., Ed., "Internet Message Format", [RFC 5322](#), October 2008.

Internet-Draft SMTP Service Extension for Message Recall

July 2009

- [IANA] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", [BCP 26](#), [RFC 5226](#), May 2008.
- [Kwds] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", [RFC 2119](#), March 1997.
- [Report] Vaudreuil, G., "The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages", [RFC 3462](#), January 2003.
- [SHA] U.S. Department of Commerce, "Secure Hash Standard", FIPS PUB 180-3, October 2008.
- [SMTP] Klensin, J., Ed., "Simple Mail Transfer Protocol", [RFC 5321](#), October 2008.

Author's Address

Barry Leiba
Huawei Technologies

Phone: +1 646 827 0648
Email: barryleiba@computer.org

Leiba

Expires January 7, 2010

[Page 17]