SMTP extensions
Internet-Draft
Obsoletes: 1845 (if approved)
Intended status: Standards Track
Expires: October 19, 2007

T. Finch
University of Cambridge

April 17, 2007

SMTP service extensions for transaction checkpointing
draft-fanf-smtp-rfc1845bis

Status of this Memo

Copyright Notice

Abstract

This memo describes the SMTP service extension for checkpoint/resume, which allows a client to recover from a lost connection to the server without having to repeat all of the commands and message content sent prior to the interruption, and with less risk of duplicated messages. It also includes an updated specification of the predecessor SMTP service extension for checkpoint/restart.

Document revision

$Cambridge: hermes/doc/qsmtp/draft-fanf-smtp-rfc1845bis.xml,v 1.76 2007/02/24 10:59:44 fanf2 Exp $

Table of Contents

1. Introduction

Although SMTP is widely and robustly deployed, it does not handle the loss of its underlying connection particularly gracefully. There are two problems, and they are becoming more of a concern because of the way the Internet is changing.

Firstly, if the connection is lost part-way through the transmission of a message, the client must retry the transmission starting from the beginning. When dealing with very large messages over less reliable connections it is possible for substantial resources to be consumed by repeated unsuccessful attempts to transmit the message in its entirety. Messages are getting larger on average. Wireless connections, which are much more likely to be lost in normal use, are becoming more common.

Secondly, a connection that is lost after the client has transmitted the message but before it receives the server's final reply can result in a duplicated message. This problem is described in [RFC1047], which recommends that servers should minimize the time between receiving the last of the message data and sending their final reply. However it is often preferable to analyse the message data for spam and viruses at this point so that the server can avoid taking responsibility for an undesirable message, and this can be time consuming.

Furthermore, the problem is worse for clients that try to make the most of high-latency connections. The minimum time for an SMTP transaction is normally one round trip, since the client has to wait for all outstanding server replies after sending the DATA command [RFC2920]. If the client's messages are smaller than the amount of data it can transmit in a round trip, i.e. the bandwidth*delay

product, then the connection is sitting idle for some of the time. A client can work around this problem by using multiple concurrent SMTP connections, or by using the SMTP service extension for large messages [RFC3030]. The latter eliminates pipeline stalls so the client can stream multiple messages to the server while waiting for replies. However, with both work-arounds, one loss of network connectivity means multiple messages will need retransmission and may be duplicated.

1.1. Overview

This memo provides a facility by which a client can uniquely identify a particular SMTP transaction. The server stores this identifying information along with all the information it receives and sends as the transaction proceeds. If the connection is lost during the transaction the SMTP client may establish a new connection and ask

the server to resume the transaction. The server tells the client how much message data it saved from the interrupted transaction. The client can then perform an abbreviated transaction, repeating only those commands to which it did not receive replies, and transmitting only message data that the server does not yet have.

These problems were originally tackled by the SMTP service extension for checkpoint/restart [RFC1845]. However it has a number of limitations (Section 3.3) so this memo describes the similar but improved SMTP service extension for checkpoint/resume in Section 2. We address backwards compatibility in Section 3 by re-defining checkpoint/restart as a modification of checkpoint/resume. Finally, Section 4 gives proper consideration to the security implications of these extensions.

1.2. Terminology

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].

The "message envelope" comprises the SMTP MAIL and RCPT commands. The "message data" is transmitted using the SMTP DATA command, or alternatives such as the BDAT command [RFC3030].

An SMTP "transaction" is the sequence of commands necessary to transmit a message, i.e. a message envelope followed by message data. It starts with a MAIL command and ends with the server's "final reply" to the last of the message data, or with a reset. The server's final reply in an LMTP (local mail transport protocol) transaction can comprise more than one per-recipient sub-reply [RFC2033]. A "reset" is caused by the RSET, HELO, or EHLO commands.

The metalinguistic notation used in this memo corresponds to the "Augmented Backus-Naur Form" defined in [RFC4234]. Rules not defined here are either defined in the ABNF core rules or in [RFC2821] or [RFC2822]. Metalanguage terms used in running text are surrounded by pointed brackets (e.g., <transid-spec>).

1.3. IANA Considerations

This memo defines the SMTP service extension for checkpoint/resume, with the EHLO keyword value "RESUME", in Section 2.

This memo replaces RFC 1845 as the definition of the SMTP service extension for checkpoint/restart, with the EHLO keyword value "CHECKPOINT", in Section 3.

2. SMTP service extension for checkpoint/resume

This section uses the SMTP extension model specified in [RFC2821].

2.1. Framework

The SMTP service extension for checkpoint/resume is defined as follows:

- The name of the service extension is "checkpoint/resume".
- The EHLO keyword associated with the extension is "RESUME".
- The RESUME EHLO keyword has no parameters.
- This extension defines one additional command, RESUME, which MAY appear anywhere in a pipelined group [RFC2920].

  Syntax:

  ```
  resume-command = "RESUME" SP transid-value CRLF
  ```

- This extension defines the 355 reply to the RESUME command.

  Syntax:

  ```
  resume-reply = "355" SP octet-offset SP text CRLF
  ```

- This extension defines two additional parameters to the MAIL command.

  The TRANSID parameter has the following syntax:

  ```
  esmtp-param     =/  transid-param
  transid-param   =   "TRANSID=" transid-value
  transid-value   =   "<" transid-spec ">"
  transid-spec    =   dot-string "@" domain
  ```

  The TRANSOFF parameter has the following syntax:

  ```
  esmtp-param     =/  transoff-param
  transoff-param  =   "TRANSOFF=" octet-offset
  octet-offset    =   1*20DIGIT
  ```

- The maximum length of the MAIL command is increased by 297 characters. The maximum length of a <transid-spec> is 256 characters.
- There are no additional parameters to the RCPT command defined by this extension and its maximum length is not increased.
- This extension is suitable for use with message submission [RFC4409] and LMTP [RFC2033].

2.2. Overview

A server that supports the SMTP service extension for checkpoint/resume SHALL include the RESUME keyword in its reply to the client's EHLO command. A client that wishes to use this extension MUST first check that this EHLO keyword is present.

The client then proceeds as usual, except that it issues MAIL commands with TRANSID and TRANSOFF parameters (Section 2.9). The TRANSID parameter's value is described in Section 2.3 and the TRANSOFF parameter's value is "0" (zero). The server periodically checkpoints the transaction, retaining state so that it can later be resumed (Section 2.5). If all goes well, the client will close the connection normally and the server can discard the resume state (Section 2.10).

If the connection is lost (Section 2.6) then the client can reconnect (Section 2.7) and issue a RESUME command (Section 2.8) for each transaction in the lost connection. It thereby discovers the <octet−offset> at which it must resume transmitting each transaction's data (Section 2.4). The client then issues MAIL commands with TRANSID and non−zero TRANSOFF parameters to resume any transactions that are missing data on the server or for which the client lost any of the server's replies.

2.3. Transaction IDs

A <transid−spec> serves to uniquely identify a particular SMTP transaction started by a particular client. The <transid−spec> and client identity together form the "transaction ID".

The <transid−spec> is structured to ensure global uniqueness without any additional registry. Its domain part SHOULD be a valid domain name that uniquely identifies the SMTP client. This is usually the same as the domain name given in the EHLO command, but not always. The EHLO domain name identifies the specific host the SMTP connection originated from, whereas the <transid−spec> domain can refer to a group of hosts that collectively host a multi−homed SMTP client, or it can refer to a mobile client's home domain name, etc.

The <transid−spec> local part MUST be an identifier that distinguishes this SMTP transaction from any other originating from

this SMTP client. Care must be used in constructing a <transid−spec> to simultaneously ensure both uniqueness, unguessability, and the ability to reidentify interrupted transactions. It MUST be unguessable for security reasons; see Section 4.

Despite the structured nature of a <transid−spec> the server MUST treat the value as an opaque, case−sensitive string.

Note that the contents of the [RFC2822] Message−ID: header field, or the MAIL FROM: ENVID parameter [RFC3461], typically are NOT appropriate for use as a <transid−spec>, since such identifiers may be associated with multiple copies of the same message − e.g., as it is split during transmission to different recipients − and hence with multiple distinct SMTP transactions.

For security reasons, servers MUST treat the same <transid−spec> from different clients as different transaction IDs. Servers MUST use a secure identifier to distinguish clients, such as credentials from the SMTP AUTH command [RFC2554] or from TLS negotiation [RFC3207]. Note that these identifiers are independent of the IP address a client connects from: servers MUST allow authenticated mobile clients to reconnect and resume transactions from different IP addresses. If a client is not authenticated the server SHOULD use its IP address to identify it.

2.4. Octet offsets

The <octet−offset> represents an offset, counting from zero, to the particular octet in the actual message data the server expects to see next. (This is also a count of how many octets the server has received and stored successfully.) This offset SHALL NOT account for the message envelope. A value of 0 would indicate that the client needs to start sending the message from the beginning, a value of 1 would indicate that the client should skip one octet, and so on, up to a maximum equal to the message size. Additional requirements apply depending on the command(s) used by the client to transmit the message data:

DATA [RFC2821] section 4.5.2: The <octet−offset> SHALL NOT count any octets added by the dot−stuffing algorithm. The offset MUST also correspond to the start of a line, i.e. equal to zero or a point immediately after a <CRLF>.

BDAT [RFC3030]: The <octet−offset> SHALL count octets in the same way as the <chunk−size> parameter to the BDAT command. It MAY point anywhere within a chunk. It SHALL NOT count any octets for the BDAT command itself.

BURL [RFC4468]: The <octet−offset> SHALL count the size of the content retrieved from the URL given in the BURL command. The offset MUST NOT correspond to a point within the BURL data; that is, the server stores the whole URL content or none of it. The <octet−offset> SHALL NOT count any octets for the BURL command itself.

These requirements are consistent with the semantics of the SIZE parameter to the MAIL command [RFC1870].

2.5. Server−side resume state

The server's "resume state" is the additional information it retains in order to implement checkpoint/resume.

The server SHOULD keep track of the transaction IDs associated with each connection, that is the <transid−spec>s used by the client in RESUME commands or TRANSID parameters. This so that it can prevent multiple connections from trying to modify the same transaction, as described in Section 2.7, and so that the server can discard the resume state of the connection's transactions when the connection is closed cleanly, as described in Section 2.10.

What is included in a transaction's resume state varies depending on whether or not the server has received any message data, and whether or not it has committed the transaction. The server commits the transaction when it decides what its final reply to the last of the message data is.

Servers retain no resume state before they receive any message data.

This implies that clients must resume the transaction from the beginning if the connection is lost while the client is transmitting the envelope.

Before the server commits a transaction, its resume state comprises the message envelope (including all client commands and server replies) and any message data that the server has received so far. It is OPTIONAL for the server to retain this state. That is, a server MAY be configured to require some or all clients to resume interrupted transactions from the beginning.

When the server has received the last of the message data it SHOULD proceed to commit the transaction regardless of whether the client connection is lost while it does so. Since committing can be a multi-stage process (especially with LMTP [RFC2033]) this ensures that the client sees a consistent result even after a connection loss and resume. If the server commits to a 2yz final reply then it can continue with onward delivery of the message independent of client activity.

After the transaction is committed, its resume state comprises the message envelope (as before), the size of the message data, and the server's final reply. If the client loses a connection and takes a while to reconnect and resume the transactions, it can be necessary for the server to retain this resume state for longer than it takes to transmit the message on to its next destination.

If the client terminates a transaction with a reset, the server SHALL discard any resume state for the transaction ID and remove the transaction ID from the list of those associated with the current connection. If the client issues a reset between transactions, the server SHALL retain any resume state for the preceding transaction(s).

2.6. Retry versus resume

A client SHOULD NOT attempt to resume a transaction unless the SMTP connection was lost after the start of the transaction. A connection is lost if it is terminated without the client sending a QUIT command to the server, or receiving a 421 reply to any command. This includes SMTP-level timeouts as well as loss or premature closure of the underlying connection. If the connection is lost before the first MAIL command, or message transmission fails for another reason, the client MUST use the retry algorithm specified by [RFC2821]. These non-resume situations include, but are not limited to, the following:

- failure to establish a connection;
- failure to establish a security layer;
- failure to authenticate;
- a temporary failure indicated by a 4yz reply from the server;
- a connection forcibly closed by the server with a 421 reply.

2.7. Re-establishing a connection

When a connection has been lost as described in the previous section, the client MAY reconnect immediately. It SHALL first re-check the DNS if necessary, as determined by the DNS records' time-to-live. If the IP address (target of A or AAAA record) used to connect to the original server is still on this list it SHOULD be tried first, since this server is most likely to be capable of resuming the transaction. If that IP address is no longer on the list or if the connection fails, then the client SHOULD next try any other IP addresses associated with the host name (target of MX record) used to connect to the original server. If these also fail then the client SHOULD fall back to the ranking algorithm described in [RFC2821] section 5.

A client SHOULD NOT wait for a server with an interrupted transaction's resume state to come back online. Multi-homed and clustered SMTP servers do exist, which means that it is entirely possible for a transaction to be resumed on a different server host.

Note that connection loss can appear different from the client and the server ends, so a client might detect the loss and reconnect before the server detects the loss. This can lead to legitimate circumstances where there are multiple connections to the server that appear to be trying to use the same transaction ID. Therefore, if a client issues a command containing a <transid-spec> that the client has used in another connection that is currently active from the server's point of view, then the server SHOULD prevent the older connection from performing further actions that affect the same transaction. For example, the server MAY drop all but the connection with the most recent activity. Even if the server is still committing a transaction when the client resumes it, the server MUST issue the final reply that it commits to.

2.8. The RESUME command

A client uses the RESUME command to discover how much message data the server has stored for a given <transid-spec>, which is specific to that client. If the server has no resume state associated with the transaction ID, it SHALL reply with a 355 code and an <octet-offset> of "0" (zero). If the server has resume state associated with the transaction ID the server SHALL reply with a 355 code and an <octet-offset> equal to the amount of message data received by the server so far.

The <octet-offset> field in a 355 reply is REQUIRED to be the first thing on the first line of the reply. It MUST be separated from any following <text> by linear whitespace. The <text> can vary and MUST be ignored by the client.

For example,

```
355 64735 is the transaction offset
```

The client MUST NOT issue a RESUME command during a MAIL transaction. The server SHOULD reject a RESUME command issued during a transaction with a "503 Bad sequence of commands" reply.

After reconnecting, clients SHOULD issue RESUME commands for all the transactions in a lost connection, not just for the transactions that were interrupted. This is so that when the client QUITs the server can discard all their resume state. See Section 2.10.

The possible failure replies to the RESUME command are 500, 501, and 421, as described in [RFC2821] section 4.3.2.

2.9. The MAIL command TRANSID and TRANSOFF parameters

A client SHALL start or resume a resumable transaction by issuing a MAIL command with one TRANSID parameter and one TRANSOFF parameter. It MUST NOT include more than one of either parameter, or omit either parameter. (However see Section 3 for the meaning of a TRANSID parameter without a TRANSOFF parameter.)

If this is a new transaction, the TRANSOFF MUST be "0" (zero). The server SHALL discard any resume state that it may have retained for the given <transid−spec>, which is specific to that client. The transaction then proceeds as normal, with the server retaining additional resume state as described in Section 2.5.

If the client is resuming a transaction, then it MUST have previously issued a RESUME command in this connection with the same <transid−spec> as in the TRANSID parameter. The value of the TRANSOFF parameter MUST be the same as the <octet−offset> given in the server's reply to the RESUME command. Apart from the non−zero <octet−offset> the MAIL command MUST be the same as the one issued by the client to start the transaction originally. If the client issues a MAIL command with a non−zero TRANSOFF that does not meet these requirements then the server SHALL reply with "503 Bad sequence of commands". Otherwise the server SHALL give the same reply to the MAIL command that it retained in the transaction's resume state.

After the client has issued a MAIL command with a non−zero TRANSOFF, it MAY re−issue the transaction's RCPT commands. The client MAY omit some or all of the RCPT commands but it MUST NOT re−order them. The server SHALL give the same reply to each RCPT command that it retained in the transaction's resume state. If the client issues a RCPT command that was not retained in the resume state then the server SHALL reject it with a "553 Requested action not taken: mailbox name not allowed" reply.

After the client has issued the envelope commands, it SHALL issue another data transfer command (e.g. DATA or BDAT) and send the remaining message data starting from the <octet−offset>. The requirement in [RFC3030] against mixing DATA and BDAT in the same transaction still applies if the transaction is interrupted and later

resumed. If the <octet−offset> is equal to the message size, then the client SHALL issue a data transfer command with no data (e.g. DATA<CRLF>.<CRLF> or BDAT 0 LAST<CRLF>) and the server SHALL give the final reply it retained in the transaction's resume state.

2.10. The QUIT command

The client MUST issue a QUIT command before closing the connection. It MUST NOT pipeline the QUIT command; that is, it SHALL wait to receive the replies to all outstanding commands before issuing the QUIT command. Note that this is stricter than [RFC2920] which allows the QUIT command to appear as the last command in a pipelined group.

This requirement means that when the server receives a QUIT command, it can be sure that the client has received all the replies to all the transactions in the connection, so the server MAY then discard any resume state associated with these transactions. The server MUST NOT rely on this for resource reclamation and MUST still time out old resume state. This protects against malicious clients and some legitimate failure modes. For example, if the connection is lost after the client sends QUIT but before the server receives it, the client will not reconnect and the server will not immediately free its resume state.

3. SMTP service extension for checkpoint/restart

This section re−defines checkpoint/restart in terms of checkpoint/resume, and lists the differences between this specification and [RFC1845]. We also explain why checkpoint/resume is better than checkpoint/restart.

3.1. Framework

The SMTP service extension for checkpoint/restart is defined as follows:

- The name of the service extension is "checkpoint/restart".
- The EHLO keyword associated with the extension is "CHECKPOINT".
- The CHECKPOINT EHLO keyword has no parameters.
- This extension defines one additional parameter to the MAIL command. The TRANSID parameter has the syntax defined in Section 2.
- The maximum length of the MAIL command is increased by 88 characters. The maximum length of a <transid−spec> is 77 characters. If the server also supports the checkpoint/resume extension, then its larger limits apply instead of (not in addition to) these limits.
- This extension defines the 355 reply to the MAIL command which has the syntax defined in Section 2.
- There are no additional parameters to the RCPT command defined by this extension and its maximum length is not increased.
- There are no additional commands defined by this extension.
- This extension is suitable for use with message submission [RFC4409].

3.2. Description

A server that supports the SMTP service extension for checkpoint/restart SHALL include the CHECKPOINT keyword in its reply to the client's EHLO command. A client that wishes to use this extension MUST first check that this EHLO keyword is present. It SHALL then issue a MAIL command with one TRANSID parameter and without a TRANSOFF parameter. Such a MAIL command MUST appear as the last command in a pipelined group; note that this is stricter than the

usual pipelining requirements for the MAIL command specified in [RFC2920].

Issuing a MAIL FROM:<...> TRANSID=<transid−spec> command to a server that supports checkpoint/restart is equivalent to issuing the following hypothetical pair of commands to a server that supports checkpoint/resume:

```
RESUME <transid−spec>
MAIL FROM:<...> TRANSID=<transid−spec> TRANSOFF=<octet−offset>
```

The <octet−offset> value of the hypothetical TRANSOFF parameter is the same as the <octet−offset> given in the server's hypothetical 355 reply to the RESUME command.

One of the two hypothetical replies to this pair of checkpoint/resume commands is given in response to the equivalent real checkpoint/restart MAIL...TRANSID command. The reply is chosen according to the following rules:

- If the hypothetical reply to the RESUME command would have indicated failure (i.e. not 355) then that is used as the real reply;
- Otherwise, if the hypothetical reply to the MAIL FROM command would have indicated failure (i.e. not 250) then that is used as the real reply;
- Otherwise, if the hypothetical <octet−offset> given by the server's 355 reply and used in the MAIL FROM command would have been zero, then the hypothetical 250 reply to the MAIL FROM command is used as the real reply; (This is the case for new transactions.)
- Otherwise, the hypothetical 355 reply to the RESUME command is used as the real reply. (This is the case for resumed transactions.)

When a client wishes to start a new resumable transaction, it SHALL issue a MAIL...TRANSID command. If it does not get the expected 250 reply (which would indicate an accidental or malicious transaction ID collision) it SHALL issue a RSET command to reset the transaction, and re−issue the MAIL...TRANSID command.

When a client wishes to resume a transaction after a lost connection, it SHALL issue the same MAIL...TRANSID command again. If it receives a 250 reply, it MUST repeat the transaction in full. If it receives a 355 reply, it MAY re−issue the rest of the transaction's envelope commands, then it SHALL issue a data transfer command and resume transmission of the message data at the exact <octet−offset> indicated in the 355 reply.

In all other respects this extension works in the same way as checkpoint/resume (Section 2).

3.3. Changes from RFC 1845

This section lists differences between checkpoint/restart as specified in this memo and as specified in [RFC1845]. We also give the reasons for each change.

- Interworking with pipelining [RFC2920] is improved in a number of ways:
  - We explicitly state that a checkpoint/restart MAIL command has stricter pipelining requirements than specified in [RFC2920].
  - A client that pipelines its envelope commands and message data (using BDAT [RFC3030] or BURL [RFC4468]) can lose the server's envelope replies when a connection is lost. Clients may now re−issue envelope commands when resuming a transaction.
  - A client can pipeline the end of one transaction with the start of the next, so a new transaction does not indicate that the previous transaction is complete from the client's point of view. Servers are no longer permitted to discard a transaction's resume state at this point.
- We now specify the semantics of the <octet−offset> when the message data is transmitted using the BDAT [RFC3030] or BURL [RFC4468] commands.
- We now specify how to use this extension with LMTP [RFC2033].
- The server's data retention requirements have been loosened. This allows a server to advertise support for checkpointing to less trustworthy clients, with less security exposure. See Section 4.
- In [RFC1845] the <transid−spec> is coupled to the client's EHLO host name. This memo specifies the use of higher−level authenticated client identities where possible, so that mobile clients can re−connect from a different IP address (which implies a different EHLO host name) and resume interrupted transactions.
- This specification allows a client to re−connect to the server sooner when resuming a transaction than is usual for normal SMTP retries, as specified in [RFC2821] section 4.5.4. This is friendlier to message submission clients that involve a human in the retry strategy.
- We have identified some security concerns that are discussed in Section 4.
- A couple of trivial matters:
  - [RFC2119] keywords for normative requirements.
  - Corrected octet counts in the MAIL command length limit increase.

3.4. Prefer checkpoint/resume to checkpoint/restart

Even after updating and clarifying [RFC1845], there remains a significant problem with checkpoint/restart, which is why this memo defines and prefers the not−quite−backwards−compatible variant checkpoint/resume.

The problem is that checkpoint/restart adds a pipeline stall to each transaction. This increases the time taken to send a message, which is particularly undesirable for message submission clients on high−

latency connections. It also prevents clients from using the BDAT command [RFC3030] to stream messages to the server without pipeline stalls. This is particularly unfortunate because losing a connection during pipelined streaming can affect multiple messages, so it is desirable to have some way of recovering the state of transactions after a lost connection.

The pipeline stall in checkpoint/restart has two functions, which checkpoint/resume handles in ways that avoid the stall.

- In checkpoint/restart, the server tells the client whether the transaction is new or is being resumed. In checkpoint/resume, this information goes from the client to the server, which removes the need for an extra round trip in normal transactions.

- When recovering from a lost connection, the client must find out the <octet−offset> at which it will resume transmitting message data. This implies a client−server round trip before resuming the transaction. In checkpoint/resume this round trip is detached from the transaction using the RESUME command. The RESUME command can be pipelined, so if multiple transactions are to be resumed only one extra round trip per connection is needed, not one per message.

4. Security considerations

The difficulties fall into three areas: additional storage requirements on the server; vulnerabilities associated with spoofed transaction IDs; and undesirable bounce messages. All are significantly mitigated if checkpoint/resume is only permitted for trustworthy clients (which generally implies secure authentication). This section also applies to [RFC1845], which does not discuss security.

4.1. Server storage

A significant difference between partial transactions and complete transactions is that the server can recover the storage used by complete transactions by delivering their messages. Thus if a server gives partial transactions a long lifetime, it can be easier for an attacker to exhaust the server's disk space than with un−extended SMTP. The attacker does not have to outrun the server's ability to process messages, nor search for a destination address to which the server cannot deliver immediately (such as an over−quota user).

The extensions in this memo permit a server to choose whether or not to store partial messages according to operational needs. For example, partial transaction storage might be permitted for authorized message submission clients, but MX clients might be required to recover from failed transactions by retrying the transaction from the start.

When partial messages are stored, their lifetime should be scaled according to the typical time it takes for a client to recover from a lost connection, which will depend on the operational environment but will often be on the order of several minutes.

Once a transaction is complete, the server still keeps some resume state so that clients can recover from connection loss during the [RFC1047] synchronization gap. Abusive clients can waste this space by failing to close connections cleanly with the QUIT command. The server has to restrict the lifetime of this resume state in a similar way to partial transactions. This resume state is relatively small, and is important for correctness (avoidance of duplicate messages) so its lifetime can be longer than that of partial messages.

This memo does not provide a mechanism for clients to discover the lifetime of partial transactions and resume state on the server.

4.2. Transaction IDs

If an attacker can guess a client's transaction IDs, it can perform a variety of attacks based on confusing a client about the state of a transaction or inserting unwanted data into a message. These attacks are made much easier by the extensions defined in this memo than in un−extended SMTP. We reduce the opportunity for attack by making transaction IDs unguessable and by tying them to the client's authenticated identity.

Resumable transactions can be used to turn a truncation attack into a message modification attack. If an attacker can cause the client's connection to break in the middle of a message, the attacker can resume the transaction and append any data to the end of the message. TCP truncation attacks are much easier than TCP modification attacks. Un−guessable transaction IDs prevent attackers that cannot sniff the client−server connection (e.g. because they are off the client−server path or because the connection has a security layer) from doing this.

Unguessability can be achieved by including enough random data. [RFC4086] provides some guidelines on how to do this securely.

An attacker that guesses a transaction ID before the client uses it could potentially append data to the start of the message. In checkpoint/resume this is prevented by the client asserting that a transaction is new. In checkpoint/restart this is prevented by requiring clients to detect this attack and reset the transaction.

As well as these client−side protections, the server has to tie transaction IDs to the client's authenticated identity. Even if an attacker can guess a transaction ID, it also has to break the client's authentication credentials in order to succeed. If it doesn't manage to do both then the server will consider the client's and attacker's transactions to have different IDs.

These extensions can be used with unauthenticated SMTP, in which case the server uses the client's IP address to identify it. However this is usually not secure, especially with wireless or dial−up connections where it is relatively easy for an attacker to steal the client's IP address.

4.3. Unnecessary bounces

It is generally preferable for SMTP servers to reject messages during the SMTP transaction instead of accepting them and later generating a bounce message. This allows message submission clients to present the error to the user immediately in a friendly manner. It also reduces the problem of backscatter from spam with bogus return paths,

assuming that spam sending software has a partial SMTP implementation that doesn't emit bounces when its spam is rejected.

The RESUME specification requires a server to honor its original replies to the RCPT commands. If the status of a recipient address changes from good to bad between the start of a transaction and its completion, it would seem that the server is forced to accept the message then generate a bounce. This should be a rare situation if we assume clients will prefer to resume transactions promptly. However it is possible for servers to reduce the problem by giving a negative reply to the end of the message data: in general a 450 reply if not all the recipients' statuses have changed, so that the client will retry the delivery later.

5. References

5.1. Normative references

[RFC2033] Myers, J., "Local Mail Transfer Protocol", RFC 2033, October 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2821] Klensin, J., "Simple Mail Transfer Protocol", RFC 2821, April 2001.

[RFC2822] Resnick, P., "Internet Message Format", RFC 2822, April 2001.

[RFC2920] Freed, N., "SMTP Service Extension for Command Pipelining", STD 60, RFC 2920, September 2000.

[RFC3030] Vaudreuil, G., "SMTP Service Extensions for Transmission of Large and Binary MIME Messages", RFC 3030, December 2000.

[RFC4234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

[RFC4468] Newman, C., "Message Submission BURL Extension", RFC 4468, May 2006.

5.2. Informative references

[RFC1047] Partridge, C., "Duplicate messages and SMTP", RFC 1047, February 1988.

[RFC1845] Crocker, D. and N. Freed, "SMTP Service Extension for Checkpoint/Restart", RFC 1845, September 1995.

[RFC1870] Klensin, J., Freed, N., and K. Moore, "SMTP Service Extension for Message Size Declaration", STD 10, RFC 1870, November 1995.

[RFC2554] Myers, J., "SMTP Service Extension for Authentication", RFC 2554, March 1999.

[RFC3207] Hoffman, P., "SMTP Service Extension for Secure SMTP over Transport Layer Security", RFC 3207, February 2002.

[RFC3461] Moore, K., "Simple Mail Transfer Protocol (SMTP) Service Extension for Delivery Status Notifications (DSNs)", RFC 3461, January 2003.

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

[RFC4409] Gellens, R. and J. Klensin, "Message Submission for Mail", RFC 4409, April 2006.

Appendix A. Acknowledgments

Thanks to Dave Crocker and Ned Freed for [RFC1845], from which this document borrows freely. Thanks are also due to Dave Cridland, Randall Gellens, and Alexey Melnikov for encouragement, review, and comments.

Appendix B. Changes since version -00

- LMTP support
- Clarified maximum value of <octet-offset>
- <octet-offset> doesn't count BURL commands
- Do not completely forbid clients from attempting to resume when there hasn't been a prior connection loss
- Clarify normal connection closure with QUIT from the client's point of view
- Resuming MAIL commands must be substantially the same as the transaction's original MAIL command
- Clarify ordering of RCPT commands in a resumed transaction
- Forbid mixing of DATA and BDAT
- Clarify failure replies to RESUME.

Appendix C. Draft discussion venue

Feedback about this draft should be emailed to the author, or to the IETF SMTP discussion mailing list, <ietf-smtp@imc.org>, or to the Lemonade working group mailing list, <lemonade@ietf.org>.

Author's Address

Tony Finch
University of Cambridge Computing Service
New Museums Site
Pembroke Street
Cambridge CB2 3QH
ENGLAND

Phone: +44 797 040 1426
Email: dot@dotat.at
URI: http://dotat.at/

Full Copyright Statement

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on−line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf−ipr@ietf.org.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).