Network Working Group
Request for Comments: 4550
Category: Standards Track

S. Maes
Oracle
A. Melnikov
Isode Ltd.

June 2006

Internet Email to Support Diverse Service Environments (Lemonade) Profile

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

This document describes a profile (a set of required extensions, restrictions, and usage modes) of the IMAP and mail submission protocols. This profile allows clients (especially those that are constrained in memory, bandwidth, processing power, or other areas) to efficiently use IMAP and Submission to access and submit mail. This includes the ability to forward received mail without needing to download and upload the mail, to optimize submission, and to efficiently resynchronize in case of loss of connectivity with the server.

The Internet Email to Support Diverse Service Environments (Lemonade) profile relies upon extensions to IMAP and Mail Submission protocols; specifically, the URLAUTH and CATENATE IMAP protocol (RFC 3501) extensions and the BURL extension to the SUBMIT protocol (RFC 4409).

Table of Contents

1. Introduction 3
1.1. Conventions Used in This Document 3
2. Forward without Download 3
2.1. Motivations 3
2.2. Message Sending Overview 4
2.3. Traditional Strategy 4
2.4. Step-by-Step Description 5
2.4.1. Message Assembly Using IMAP CATENATE Extension 6
2.4.2. Message Assembly Using SMTP CHUNKING and BURL Extensions 10
2.5. Normative Statements Related to Forward without Download ..14
2.6. Security Considerations for "pawn-tickets" 14
2.7. The fcc Problem 15
2.8. Registration of $Forwarded IMAP Keyword 15
3. Message Submission 15
3.1. Pipelining 16
3.2. DSN Support 16
3.3. Message Size Declaration 16
3.4. Enhanced Status Code Support 16
3.5. TLS 16
4. Quick Resynchronization 16
5. Additional IMAP Extensions 17
6. Summary of the Required IMAP and SMTP Extensions 17
7. Future work 18
8. Security Considerations 18
8.1. Confidentiality Protection of Submitted Messages 19
8.2. TLS 19
9. References 20
9.1. Normative References 20
9.2. Informative References 21
10. Acknowledgements 21

1. Introduction

Lemonade provides enhancements to Internet email to support diverse service environments.

This document describes the Lemonade profile, which includes:

- "forward without download", which describes exchanges between Lemonade clients and servers to allow new email messages to be submitted incorporating content that resides on locations external to the client.

- Quick mailbox resynchronization using [CONDSTORE].

- Several IMAP and SMTP extensions that save bandwidth and/or number of round-trips required to send/receive data.

The organization of this document is as follows. Section 2 describes "forward without download". Section 3 describes additional SMTP extensions that must be supported by all Lemonade Submission servers. Section 4 describes IMAP quick resynchronization.

1.1. Conventions Used in This Document

In examples, "M:", "I:", and "S:" indicate lines sent by the client messaging user agent, IMAP e-mail server, and SMTP submit server, respectively.

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].

All examples in this document are optimized for Lemonade use and might not represent examples of proper protocol usage for a general use Submit/IMAP client. In particular, examples assume that Lemonade Submit and IMAP servers support all Lemonade extensions described in this document, so they don't show how to deal with absence of an extension.

2. Forward without Download

2.1. Motivations

The advent of client/server email using the [RFC3501], [RFC2821], and [SUBMIT] protocols has changed what formerly were local disk operations into repetitive network data transmissions.

Lemonade "forward without download" makes use of the [BURL] SUBMIT extension to enable access to external sources during the submission of a message. In combination with the IMAP [URLAUTH] extension, inclusion of message parts or even entire messages from the IMAP mail store is possible with a minimal trust relationship between the IMAP and SMTP SUBMIT servers.

Lemonade "forward without download" has the advantage of maintaining one submission protocol, and thus avoids the risk of having multiple parallel and possibly divergent mechanisms for submission. The client can use Submit/SMTP [SUBMIT] extensions without these being added to IMAP. Furthermore, by keeping the details of message submission in the SMTP SUBMIT server, Lemonade "forward without download" can work with other message retrieval protocols such as Post Office Protocol (POP), Network News Transfer Protocol (NNTP), or whatever else may be designed in the future.

2.2. Message Sending Overview

The act of sending an email message can be thought of as involving multiple steps: initiation of a new draft, draft editing, message assembly, and message submission.

Initiation of a new draft and draft editing takes place in the Mail User Agent (MUA). Frequently, users choose to save more complex messages on an [RFC3501] server (via the APPEND command with the \Draft flag) for later recall by the MUA and resumption of the editing process.

Message assembly is the process of producing a complete message from the final revision of the draft and external sources. At assembly time, external data is retrieved and inserted in the message.

Message submission is the process of inserting the assembled message into the [RFC2821] infrastructure, typically using the [SUBMIT] protocol.

2.3. Traditional Strategy

Traditionally, messages are initiated, edited, and assembled entirely within an MUA, although drafts may be saved to an [RFC3501] server and later retrieved from the server. The completed text is then transmitted to a Message Submission Agent (MSA) for delivery.

There is often no clear boundary between the editing and assembly process. If a message is forwarded, its content is often retrieved immediately and inserted into the message text. Similarly, when external content is inserted or attached, the content is usually retrieved immediately and made part of the draft.

As a consequence, each save of a draft and subsequent retrieve of the draft transmits that entire (possibly large) content, as does message submission.

In the past, this was not much of a problem, because drafts, external data, and the message submission mechanism were typically located on the same system as the MUA. The most common problem was running out of disk quota.

2.4. Step-by-Step Description

The model distinguishes among a Mail User Agent (MUA), an IMAP4Rev1 Server ([RFC3501]), and a SMTP submit server ([SUBMIT]), as illustrated in Figure 1.

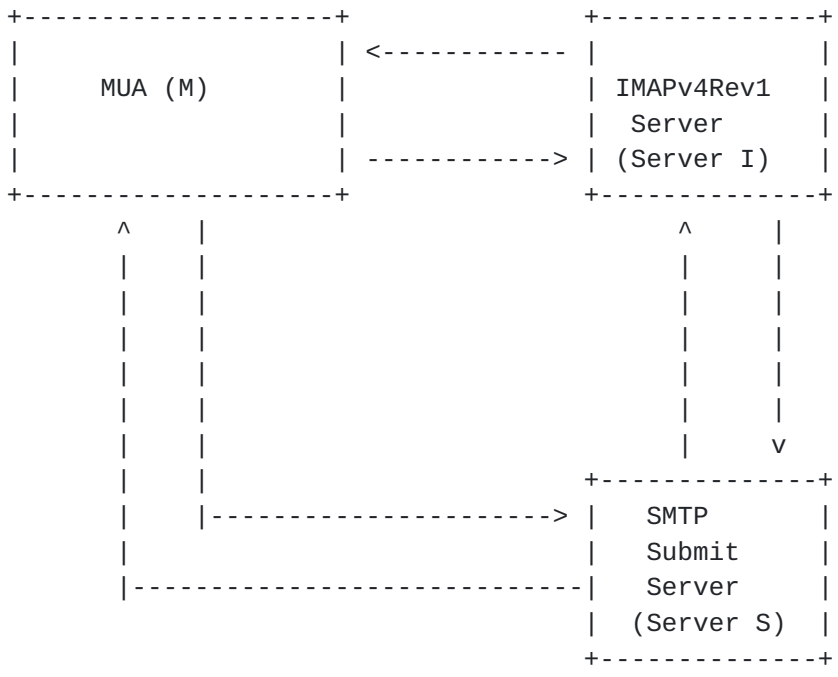

Figure 1: Lemonade "forward without download"

Lemonade "forward without download" allows a Messaging User Agent to compose and forward an e-mail combining fragments that are located in an IMAP server, without having to download these fragments to the client.

There are two ways to perform "forward without download", based on where the message assembly takes place. The first uses an extended APPEND command [CATENATE] to edit a draft message in the message store and cause the message assembly on the IMAP server. The second uses a succession of BURL and BDAT commands to submit and assemble (through concatenation) message data from the client and external data fetched from the provided URL. The two subsequent sections provide step-by-step instructions on how "forward without download" is achieved.

2.4.1. Message Assembly Using IMAP CATENATE Extension

In the [BURL]/[CATENATE] variant of the Lemonade "forward without download" strategy, messages are initially composed and edited within an MUA. The [CATENATE] extension to [RFC3501] is then used to create the messages on the IMAP server by transmitting new text and assembling them. The [UIDPLUS] IMAP extension is used by the client in order to learn the Unique Identifier (UID) of the created messages. Finally, a [URLAUTH] format URL is given to a [SUBMIT] server for submission using the [BURL] extension.

The flow involved to support such a use case consists of:

M: {to I -- Optional} The client connects to the IMAP server, optionally starts TLS (if data confidentiality is required), authenticates, opens a mailbox ("INBOX" in the example below) and fetches body structures (See [RFC3501]).

Example:

```
M: A0051 UID FETCH 25627 (UID BODYSTRUCTURE)
I: * 161 FETCH (UID 25627 BODYSTRUCTURE (("TEXT" "PLAIN"
   ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23)(
   "TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME"
   "trip.txt")
   "<960723163407.20117h@washington.example.com>"
   "Your trip details" "BASE64" 4554 73) "MIXED"))
I: A0051 OK completed
```

M: {to I} The client invokes CATENATE (See [CATENATE] for details of the semantics and steps) -- this allows the MUA to create messages on the IMAP server using new data combined with one or more message parts already present on the IMAP server.

Note that the example for this step doesn't use the LITERAL+ [LITERAL+] extension. Without LITERAL+, the new message is constructed using 3 round-trips. If LITERAL+ is used, the new message can be constructed using one round-trip.

```
   M: A0052 APPEND Sent FLAGS (\Seen $MDNSent)
      CATENATE (TEXT {475}
   I: + Ready for literal data
   M: Message-ID: <419399E1.6000505@caernarfon.example.org>
   M: Date: Thu, 12 Nov 2004 16:57:05 +0000
   M: From: Bob Ar <bar@example.org>
   M: MIME-Version: 1.0
   M: To: foo@example.net
   M: Subject: About our holiday trip
   M: Content-Type: multipart/mixed;
   M:     boundary="------------030308070208000400050907"
   M:
   M: --------------030308070208000400050907
   M: Content-Type: text/plain; format=flowed
   M:
   M: Our travel agent has sent the updated schedule.
   M:
   M: Cheers,
   M: Bob
   M: --------------030308070208000400050907
   M:  URL "/INBOX;UIDVALIDITY=385759045/;
      UID=25627/;Section=2.MIME" URL "/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2" TEXT {44}
   I: + Ready for literal data
   M:
   M: --------------030308070208000400050907--
   M: )
   I: A0052 OK [APPENDUID 387899045 45] CATENATE Completed
```

M: {to I} The client uses GENURLAUTH command to request a URLAUTH URL (see [URLAUTH]).

I: {to M} The IMAP server returns a URLAUTH URL suitable for later retrieval with URLFETCH (see [URLAUTH] for details of the semantics and steps).

```
   M: A0054 GENURLAUTH "imap://bob.ar@example.org/Sent;
      UIDVALIDITY=387899045/;uid=45;expire=2005-10-
      28T23:59:59Z;urlauth=submit+bob.ar" INTERNAL
   I: * GENURLAUTH "imap://bob.ar@example.org/Sent;
      UIDVALIDITY=387899045/;uid=45;expire=
      2005-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:91354a473744909de610943775f92038"
   I: A0054 OK GENURLAUTH completed
```

M: {to S} The client connects to the mail submission server and starts a new mail transaction. It uses BURL to let the SMTP submit server fetch the content of the message from the IMAP

server. (See [BURL] for details of the semantics and steps.) This allows the MUA to authorize the SMTP submit server to access the message composed as a result of the CATENATE step. Note that the second EHLO command is required after a successful STARTTLS command. Also note that there might be a third required EHLO command if the second EHLO response doesn't list any BURL options. Section 2.4.2 demonstrates this.

```
S: 220 owlry.example.org ESMTP
M: EHLO potter.example.org
S: 250-owlry.example.com
S: 250-8BITMIME
S: 250-BINARYMIME
S: 250-PIPELINING
S: 250-BURL imap
S: 250-CHUNKING
S: 250-AUTH PLAIN
S: 250-DSN
S: 250-SIZE 10240000
S: 250-STARTTLS
S: 250 ENHANCEDSTATUSCODES
M: STARTTLS
S: 220 Ready to start TLS
...TLS negotiation, subsequent data is encrypted...
M: EHLO potter.example.org
S: 250-owlry.example.com
S: 250-8BITMIME
S: 250-BINARYMIME
S: 250-PIPELINING
S: 250-BURL imap
S: 250-CHUNKING
S: 250-AUTH PLAIN
S: 250-DSN
S: 250-SIZE 10240000
S: 250 ENHANCEDSTATUSCODES
M: AUTH PLAIN aGFycnkAaGFycnkAYWNjaW8=
S: 235 2.7.0 PLAIN authentication successful.
M: MAIL FROM:<bob.ar@example.org>
S: 250 2.5.0 Address Ok.
M: RCPT TO:<foo@example.net>
S: 250 2.1.5 foo@example.net OK.
M: BURL imap://bob.ar@example.org/Sent;UIDVALIDITY=387899045/;
   uid=45/;urlauth=submit+bar:internal:
   91354a473744909de610943775f92038 LAST
```

S: {to I} The mail submission server uses URLFETCH to fetch the message to be sent. (See [URLAUTH] for details of the semantics and steps. The so-called "pawn-ticket" authorization mechanism uses a URI that contains its own authorization credentials.)

I: {to S} Provides the message composed as a result of the CATENATE step.

Mail submission server opens IMAP connection to the IMAP server:

```
I: * OK [CAPABILITY IMAP4REV1 STARTTLS NAMESPACE LITERAL+
   CATENATE URLAUTH UIDPLUS CONDSTORE IDLE] imap.example.com
   IMAP server ready
S: a000 STARTTLS
I: a000 Start TLS negotiation now
...TLS negotiation, if successful - subsequent data
   is encrypted...
S: a001 LOGIN submitserver secret
I: a001 OK submitserver logged in
S: a002 URLFETCH "imap://bob.ar@example.org/Sent;
   UIDVALIDITY=387899045/;uid=45/;urlauth=submit+bob.ar:
   internal:91354a473744909de610943775f92038"
I: * URLFETCH "imap://bob.ar@example.org/Sent;
   UIDVALIDITY=387899045/;uid=45/;urlauth=submit+bob.ar:
   internal:91354a473744909de610943775f92038" {15065}
...message body follows...
S: a002 OK URLFETCH completed
I: a003 LOGOUT
S: * BYE See you later
S: a003 OK Logout successful
```

Note that if the IMAP server doesn't send CAPABILITY response code in the greeting, the mail submission server must issue the CAPABILITY command to learn about supported IMAP extensions as described in RFC 3501.

Also, if data confidentiality is not required, the mail submission server may omit the STARTTLS command before issuing the LOGIN command.

S: {to M} Submission server assembles the complete message, and if the assembly succeeds, it returns OK to the MUA:

```
S: 250 2.5.0 Ok.
```

M: {to I} The client marks the message containing the forwarded attachment on the IMAP server.

```
M: A0053 UID STORE 25627 +FLAGS.SILENT ($Forwarded)
I: * 215 FETCH (UID 25627 MODSEQ (12121231000))
I: A0053 OK STORE completed
```

Note: the UID STORE command shown above will only work if the marked message is in the currently selected mailbox; otherwise, it requires a SELECT. This command can be omitted. The untagged FETCH response is due to [CONDSTORE]. The $Forwarded IMAP keyword is described in Section 2.8.

2.4.2. Message Assembly Using SMTP CHUNKING and BURL Extensions

In the [BURL]/[CHUNKING] variant of the Lemonade "forward without download" strategy, messages are initially composed and edited within an MUA. During submission [SUBMIT], BURL [BURL] and BDAT [CHUNKING] commands are used to create the messages from multiple parts. New body parts are supplied using BDAT commands, while existing body parts are referenced using [URLAUTH] format URLs in BURL commands.

The flow involved to support such a use case consists of:

M: {to I -- Optional} The client connects to the IMAP server, optionally starts TLS (if data confidentiality is required), authenticates, opens a mailbox ("INBOX" in the example below), and fetches body structures (see [RFC3501]).

Example:

```
M: A0051 UID FETCH 25627 (UID BODYSTRUCTURE)
I: * 161 FETCH (UID 25627 BODYSTRUCTURE (("TEXT" "PLAIN"
   ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23)(
   "TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME"
   "trip.txt")
   "<960723163407.20117h@washington.example.com>"
   "Your trip details" "BASE64" 4554 73) "MIXED"))
I: A0051 OK completed
```

M: {to I} The client uses GENURLAUTH command to request URLAUTH URLs (see [URLAUTH]) referencing pieces of the message to be assembled.

I: {to M} The IMAP server returns URLAUTH URLs suitable for later retrieval with URLFETCH (see [URLAUTH] for details of the semantics and steps).

```
M: A0054 GENURLAUTH "imap://bob.ar@example.org/INBOX;
   UIDVALIDITY=385759045/;UID=25627/;Section=2.MIME;
   expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar"
```

```
      INTERNAL "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar" INTERNAL
   I: * GENURLAUTH "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2.MIME;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:A0DEAD473744909de610943775f9BEEF"
      "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:BEEFA0DEAD473744909de610943775f9"
   I: A0054 OK GENURLAUTH completed
```

M: {to S} The client connects to the mail submission server and starts a new mail transaction. It uses BURL to instruct the SMTP submit server to fetch from the IMAP server pieces of the message to be sent (see [BURL] for details of the semantics and steps). Note that the second EHLO command is required after a successful STARTTLS command. The third EHLO command is required if and only if the second EHLO response doesn't list any BURL options. See Section 2.4.1 for an example of submission where the third EHLO command/response is not present.

```
   S: 220 owlry.example.org ESMTP
   M: EHLO potter.example.org
   S: 250-owlry.example.com
   S: 250-8BITMIME
   S: 250-BINARYMIME
   S: 250-PIPELINING
   S: 250-BURL
   S: 250-CHUNKING
   S: 250-AUTH DIGEST-MD5
   S: 250-DSN
   S: 250-SIZE 10240000
   S: 250-STARTTLS
   S: 250 ENHANCEDSTATUSCODES
   M: STARTTLS
   S: 220 Ready to start TLS
   ...TLS negotiation, subsequent data is encrypted...
   M: EHLO potter.example.org
   S: 250-owlry.example.com
   S: 250-8BITMIME
   S: 250-BINARYMIME
   S: 250-PIPELINING
   S: 250-BURL
   S: 250-CHUNKING
   S: 250-AUTH DIGEST-MD5 CRAM-MD5 PLAIN EXTERNAL
   S: 250-DSN
```

```
S: 250-SIZE 10240000
S: 250 ENHANCEDSTATUSCODES
M: AUTH PLAIN aGFycnkAaGFycnkAYWNjaW8=
S: 235 2.7.0 PLAIN authentication successful.
M: EHLO potter.example.org
S: 250-owlry.example.com
S: 250-8BITMIME
S: 250-BINARYMIME
S: 250-PIPELINING
S: 250-BURL imap imap://imap.example.org
S: 250-CHUNKING
S: 250-AUTH DIGEST-MD5 CRAM-MD5 PLAIN EXTERNAL
S: 250-DSN
S: 250-SIZE 10240000
S: 250 ENHANCEDSTATUSCODES
M: MAIL FROM:<bob.ar@example.org> BODY=BINARY
S: 250 2.5.0 Address Ok.
M: RCPT TO:<foo@example.net>
S: 250 2.1.5 foo@example.net OK.
M: BDAT 475
M: Message-ID: <419399E1.6000505@caernarfon.example.org>
M: Date: Thu, 12 Nov 2004 16:57:05 +0000
M: From: Bob Ar <bar@example.org>
M: MIME-Version: 1.0
M: To: foo@example.net
M: Subject: About our holiday trip
M: Content-Type: multipart/mixed;
M:     boundary="------------030308070208000400050907"
M:
M: --------------030308070208000400050907
M: Content-Type: text/plain; format=flowed
M:
M: Our travel agent has sent the updated schedule.
M:
M: Cheers,
M: Bob
M: --------------030308070208000400050907
S: 250 2.5.0 OK
M: BURL imap://bob.ar@example.org/INBOX;
   UIDVALIDITY=385759045/;UID=25627/;Section=2.MIME;
   expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
   internal:A0DEAD473744909de610943775f9BEEF
S: 250 2.5.0 OK
M: BURL imap://bob.ar@example.org/INBOX;
   UIDVALIDITY=385759045/;UID=25627/;Section=2;
   expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
   internal:BEEFA0DEAD473744909de610943775f9
S: 250 2.5.0 OK
```

```
   M: BDAT 44 LAST
   M:
   M: --------------030308070208000400050907--
```

S: {to I} The mail submission server uses URLFETCH to fetch the pieces of the message to be sent (see [URLAUTH] for details of the semantics and steps). The so-called "pawn-ticket" authorization mechanism uses a URI that contains its own authorization credentials.

I: {to S} Returns the requested body parts.

Mail submission server opens IMAP connection to the IMAP server:

```
   I: * OK [CAPABILITY IMAP4REV1 STARTTLS NAMESPACE LITERAL+
      CATENATE URLAUTH UIDPLUS CONDSTORE IDLE] imap.example.com
      IMAP server ready
   S: a001 LOGIN submitserver secret
   I: a001 OK submitserver logged in
   S: a002 URLFETCH "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2.MIME;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:A0DEAD473744909de610943775f9BEEF" "imap://
      bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:BEEFA0DEAD473744909de610943775f9"
   I: * URLFETCH "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2.MIME;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:A0DEAD473744909de610943775f9BEEF" {84}
   ...message section follows...
       "imap://bob.ar@example.org/INBOX;
      UIDVALIDITY=385759045/;UID=25627/;Section=2;
      expire=2006-10-28T23:59:59Z;urlauth=submit+bob.ar:
      internal:BEEFA0DEAD473744909de610943775f9" {15065}
   ...message section follows...
   S: a002 OK URLFETCH completed
   I: a003 LOGOUT
   S: * BYE See you later
   S: a003 OK Logout successful
```

Note that if the IMAP server doesn't send CAPABILITY response code in the greeting, the mail submission server must issue the CAPABILITY command to learn about supported IMAP extensions as described in RFC 3501.

Also, if data confidentiality is required, the mail submission server should start TLS before issuing the LOGIN command.

S: {to M} Submission server assembles the complete message, and if the assembly succeeds, it acknowledges acceptance of the message by sending 250 response to the last BDAT command:

```
S: 250 2.5.0 Ok, message accepted.
```

M: {to I} The client marks the message containing the forwarded attachment on the IMAP server.

```
M: A0053 UID STORE 25627 +FLAGS.SILENT ($Forwarded)
I: * 215 FETCH (UID 25627 MODSEQ (12121231000))
I: A0053 OK STORE completed
```

Note: the UID STORE command shown above will only work if the marked message is in the currently selected mailbox; otherwise, it requires a SELECT. This command can be omitted. The untagged FETCH response is due to [CONDSTORE]. The $Forwarded IMAP keyword is described in Section 2.8.

2.5. Normative Statements Related to Forward without Download

Lemonade-compliant IMAP servers MUST support IMAP4Rev1 [RFC3501], CATENATE [CATENATE], UIDPLUS [UIDPLUS], and URLAUTH [URLAUTH]. This support MUST be declared via CAPABILITY [RFC3501].

Lemonade-compliant submit servers MUST support BURL [BURL], 8BITMIME [8BITMIME], BINARYMIME [CHUNKING], and CHUNKING [CHUNKING]. This support MUST be declared via EHLO [RFC2821]. BURL MUST support URLAUTH type URLs [URLAUTH], and thus MUST advertise the "imap" option following the BURL EHLO keyword (see [BURL] for more details).

Additional normative statements are provided in other sections.

2.6. Security Considerations for "pawn-tickets"

The so-called "pawn-ticket" authorization mechanism uses a URI, which contains its own authorization credentials using [URLAUTH]. The advantage of this mechanism is that the SMTP submit [SUBMIT] server cannot access any data on the [RFC3501] server without a "pawn-ticket" created by the client.

The "pawn-ticket" grants access only to the specific data that the SMTP submit [SUBMIT] server is authorized to access, can be revoked by the client, and can have a time-limited validity.

2.7. The fcc Problem

The "fcc problem" refers to delivering a copy of a message to a "file carbon copy" recipient. By far, the most common case of fcc is a client leaving a copy of outgoing mail in a "Sent Mail" or "Outbox" mailbox.

In the traditional strategy, the MUA duplicates the effort spent in transmitting to the MSA by writing the message to the fcc destination in a separate step. This may be a write to a local disk file or an APPEND to a mailbox on an IMAP server. The latter is one of the "repetitive network data transmissions" that represents the "problem" aspect of the "fcc problem".

The [CATENATE] extension to [RFC3501] can be used to address the fcc problem. The final message is constructed in the mailbox designed for outgoing mail. Note that the [CATENATE] extension can only create a single message and only on the server that stages the outgoing message for submission. Additional copies of the message can be created on the same server using one or more COPY commands.

2.8. Registration of $Forwarded IMAP Keyword

The $Forwarded IMAP keyword is used by several IMAP clients to specify that the message was resent to another email address, embedded within or attached to a new message. A mail client sets this keyword when it successfully forwards the message to another email address. Typical usage of this keyword is to show a different (or additional) icon for a message that has been forwarded. Once set, the flag SHOULD NOT be cleared.

Lemonade-compliant servers MUST be able to store the $Forwarded keyword. They MUST preserve it on the COPY operation. The servers MUST support the SEARCH KEYWORD $Forwarded.

3. Message Submission

Lemonade-compliant mail submission servers are expected to implement the following set of SMTP extensions to make message submission efficient.

Lemonade clients should take advantage of these features.

3.1. Pipelining

Mobile clients regularly use networks with a relatively high latency. Avoidance of round-trips within a transaction has a great advantage for reduction in both bandwidth and total transaction time. For this reason, Lemonade-compliant mail submission servers MUST support the SMTP Service Extensions for Command Pipelining [RFC2920].

Clients SHOULD pipeline SMTP commands when possible.

3.2. DSN Support

Lemonade-compliant mail submission servers MUST support SMTP service extensions for delivery status notifications [RFC3461].

3.3. Message Size Declaration

Lemonade-compliant mail submission servers MUST support the SMTP Service Extension for Message Size Declaration [RFC1870].

Lemonade-compliant mail submission servers MUST "expand" all BURL parts before enforcing a message size limit.

A Lemonade-compliant client SHOULD use message size declaration. In particular, it MUST NOT send a message to a mail submission server, if the client knows that the message exceeds the maximal message size advertised by the submission server.

3.4. Enhanced Status Code Support

Lemonade-compliant mail submission servers MUST support SMTP Service Extension for Returning Enhanced Error Codes [RFC2034].

3.5. TLS

Lemonade-compliant mail submission servers MUST support SMTP Service Extension for Secure SMTP over TLS [SMTP-TLS].

4. Quick Resynchronization

Lemonade-compliant IMAP servers MUST support the CONDSTORE [CONDSTORE] extension. It allows a client to quickly resynchronize any mailbox by asking the server to return all flag changes that have occurred since the last known mailbox synchronization mark.

[IMAP-DISC] shows how to perform quick mailbox resynchronization.

5. Additional IMAP Extensions

Lemonade-compliant IMAP servers MUST support the NAMESPACE [NAMESPACE] extension. The extension allows clients to discover shared mailboxes and mailboxes belonging to other users.

Lemonade-compliant IMAP servers MUST support the LITERAL+ [LITERAL+] extension. The extension allows clients to save a round-trip each time a non-synchronizing literal is sent.

Lemonade-compliant IMAP servers MUST support the IDLE [IDLE] extension. The extension allows clients to receive instant notifications about changes in the selected mailbox, without needing to poll for changes.

Lemonade-compliant IMAP servers MUST support IMAP over TLS [RFC3501] as required by RFC 3501.

6. Summary of the Required IMAP and SMTP Extensions

Name of SMTP extension	Comment
PIPELINING	Section 3.1
DSN	Section 3.2
SIZE	Section 3.3
ENHANCEDSTATUSCODES	Section 3.4
STARTTLS	Section 3.5
BURL	Forward without download, Section 2
URLAUTH support in BURL	Section 2.5
CHUNKING, BINARYMIME	Section 2.5 Section 2.5
8BITMIME,	Required by BURL
AUTH	Required by Submission, See [SMTPAUTH].

Name of IMAP extension or feature	Comment
NAMESPACE	Section 5
CONDSTORE	Section 4
STARTTLS	Required by IMAP (RFC3501)
URLAUTH, CATENATE, UIDPLUS	Forward without download, Section 2
LITERAL+	Section 5
IDLE	Section 5
$Forwarded IMAP keyword	Section 2.8

7. Future work

The Lemonade Working Group is looking into additional issues related to usage of email by mobile devices, possibly including:

- Media conversion (static and possibly streamed)
- Transport optimization for low or costly bandwidth and less reliable mobile networks (e.g., quick reconnect)
- Server to client notifications, possibly outside of the traditional IMAP band
- Dealing with firewall and intermediaries
- Compression and other bandwidth optimization
- Filtering
- Other considerations for mobile clients

8. Security Considerations

Security considerations on Lemonade "forward without download" are discussed throughout Section 2. Additional security considerations can be found in [RFC3501] and other documents describing other SMTP and IMAP extensions comprising the Lemonade profile.

Note that the mandatory-to-implement authentication mechanism for SMTP submission is described in [SUBMIT]. The mandatory-to-implement authentication mechanism for IMAP is described in [RFC3501].

8.1. Confidentiality Protection of Submitted Messages

When clients submit new messages, link protection such as TLS guards against an eavesdropper seeing the contents of the submitted message. It's worth noting, however, that even if TLS is not used, the security risks are no worse if BURL is used to reference the text than if the text is submitted directly. If BURL is not used, an eavesdropper gains access to the full text of the message. If BURL is used, the eavesdropper may or may not be able to gain such access, depending on the form of BURL used. For example, some forms restrict use of the URL to an entity authorized as a submission server or a specific user.

8.2. TLS

When Lemonade clients use the BURL extension to mail submission, which requires sending a URLAUTH token to the mail submission server, such a token should be protected from interception to avoid a replay attack that may disclose the contents of the message to an attacker. TLS-based encryption of the mail submission path will provide protection against this attack.

Lemonade clients SHOULD use TLS-protected IMAP and mail submission channels when using BURL-based message submission to protect the URLAUTH token from interception.

Lemonade-compliant mail submission servers SHOULD use TLS-protected IMAP connections when fetching message content using the URLAUTH token provided by the Lemonade client.

When a client uses SMTP STARTTLS to send a BURL command that references non-public information, there is a user expectation that the entire message content will be treated confidentially. To meet this expectation, the message submission server should use STARTTLS or a mechanism providing equivalent data confidentiality when fetching the content referenced by that URL.

9. References

9.1. Normative References

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

[8BITMIME] Klensin, J., Freed, N., Rose, M., Stefferud, E., and D. Crocker, "SMTP Service Extension for 8bit-MIMEtransport", RFC 1652, July 1994.

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

[CATENATE] Resnick, P., "Internet Message Access Protocol (IMAP) CATENATE Extension", RFC 4469, April 2006.

[UIDPLUS] Crispin, M., "Internet Message Access Protocol (IMAP) - UIDPLUS extension", RFC 4315, December 2005.

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

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

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

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

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

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

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

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

[URLAUTH] Crispin, M., "Internet Message Access Protocol (IMAP) - URLAUTH Extension", RFC 4467, May 2006.

[RFC2034] Freed, N., "SMTP Service Extension for Returning Enhanced Error Codes", RFC 2034, October 1996.

[NAMESPACE] Gahrns, M. and C. Newman, "IMAP4 Namespace", RFC 2342, May 1998.

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

[LITERAL+] Myers, J., "IMAP4 non-synchronizing literals", RFC 2088, January 1997.

[CONDSTORE] Melnikov, A. and S. Hole, "IMAP Extension for Conditional STORE Operation or Quick Flag Changes Resynchronization", RFC 4551, June 2006.

[IDLE] Leiba, B., "IMAP4 IDLE command", RFC 2177, June 1997.

9.2. Informative References

[IMAP-DISC] Melnikov, A., "Synchronization operations for disconnected IMAP4 clients", Work in Progress, October 2004.

10. Acknowledgements

This document is a product of Lemonade WG. The editors thank the Lemonade WG members that contributed comments and corrections; in particular: Randy Gellens, Dave Cridland, and Greg Vaudreuil.

This document borrows some text from "Message Submission" (February 2004) by Mark Crispin, as well as from the trio [BURL], [CATENATE], and [URLAUTH].

Authors' Addresses

Stephane H. Maes
Oracle Corporation
500 Oracle Parkway
M/S 4op634
Redwood Shores, CA 94065
USA

Phone: +1-650-607-6296
EMail: stephane.maes@oracle.com

Alexey Melnikov
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex
TW12 2BX
UK

EMail: Alexey.melnikov@isode.com

Full Copyright Statement

Intellectual Property

Acknowledgement

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