LEMONADE
Internet-Draft
Obsoletes: 4550 (if approved)
Intended status: Standards Track
Expires: August 27, 2009

D. Cridland, Ed.
A. Melnikov, Ed.
Isode Limited
S. Maes, Ed.
Oracle

February 23, 2009

The Lemonade Profile
draft-ietf-lemonade-profile-bis-12.txt

Abstract

This document describes a profile (a set of required extensions, restrictions and usage modes), dubbed Lemonade, of the IMAP, mail submission and Sieve 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 Lemonade profile relies upon several extensions to IMAP and Mail Submission protocols.

Table of Contents

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 [KEYWORDS].

Other capitalised words are typically names of extensions or commands - these are uppercased for clarity only, and are case-insensitive.

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 Submit and IMAP servers support all Lemonade extensions described in this document, so they do not demonstrate fallbacks in the absence of an extension.

2. Introduction

The Lemonade Profile, or simply Lemonade, provides enhancements to Internet email to support diverse service environments. Lemonade Mail Servers provide both a Lemonade Submission Server and a Lemonade Message Store, which are based on the existing [SUBMIT] and [IMAP] protocols respectively. They MAY also include a Lemonade Message Delivery Agent, which provides delivery-time filtering services based on [SIEVE].

This document describes the Lemonade profile that includes:

- General common enhancements to Internet Mail, described in Section 5 and Section 4.
- "Forward without download" that describes exchanges between Lemonade clients and servers to allow to submit new email messages incorporating content which resides on locations external to the client, described in Section 8.
- Quick mailbox resynchronization, described in Section 5.1.
- Extensions to support more precise, and broader, notifications from the store in support of notifications and view filters, described in Section 5.4.1 and Section 5.5.
- Delivery-time filtering in support of typical mail management use-cases, as described in Section 3.3.

The LEMONADE WG used the architecture shown in [LEMONADE-ARCH] to develop the Lemonade Profile.

It is intended that the Lemonade profile support realizations of the OMA's mobile email enabler (MEM) (see [OMA-MEM-REQ] and [OMA-MEM-ARCH]) using Internet Mail protocols defined by the IETF.

3. Summary of the required support

3.1. Lemonade Submission Servers

Lemonade Submission Servers MUST provide a service as described in [SUBMIT], and MUST support the following. Note that the Lemonade Profile imposes further requirements for some cases, detailed in the sections cited.

SMTP extension	Reference	Requirements
8BITMIME	[SMTP-8BITMIME]	[SMTP-BURL]
AUTH	[SMTP-AUTH]	[SUBMIT]
BINARYMIME	[SMTP-BINARYMIME]	Section 4.1
BURL imap	[SMTP-BURL]	Section 8
CHUNKING	[SMTP-BINARYMIME]	Section 4.1
DSN	[SMTP-DSN]	Section 4.3
ENHANCEDSTATUSCODES	[SMTP-STATUSCODES]	Section 4.5
PIPELINING	[SMTP-PIPELINING]	Section 4.2
SIZE	[SMTP-SIZE]	Section 4.4
STARTTLS	[SMTP-TLS]	Section 4.6

3.2. Lemonade Message Stores

Lemonade Message Stores MUST provide a service as described in [IMAP], and MUST support the following. Note that the Lemonade Profile imposes further requirements for some cases, detailed in the sections cited.

IMAP extension	Reference	Requirements
BINARY	[IMAP-BINARY]	Section 5.2
CATENATE	[IMAP-CATENATE]	Section 5.7
COMPRESS=DEFLATE	[IMAP-COMPRESS]	Section 5.3
CONDSTORE	[IMAP-CONDSTORE]	Section 5.1
CONTEXT=SEARCH	[IMAP-CONTEXT]	Section 5.5
CONTEXT=SORT	[IMAP-CONTEXT]	Section 5.5
CONVERT	[IMAP-CONVERT]	Section 5.2
ENABLE	[IMAP-ENABLE]	Section 5.1
ESEARCH	[IMAP-ESEARCH]	Section 5.5
ESORT	[IMAP-CONTEXT]	Section 5.5
I18NLEVEL=1	[IMAP-I18N]	Section 5.8
IDLE	[IMAP-IDLE]	Section 5.4.1
LITERAL+	[IMAP-LITERAL+]	Section 5.8
NAMESPACE	[IMAP-NAMESPACE]	Section 5.6
NOTIFY	[IMAP-NOTIFY]	Section 5.4.1
QRESYNC	[IMAP-QRESYNC]	Section 5.1
SASL-IR	[IMAP-SASL-IR]	Section 5.8
SORT	[IMAP-SORT]	Section 5.5
STARTTLS	[IMAP]	-
UIDPLUS	[IMAP-UIDPLUS]	Section 5.7
URLAUTH	[IMAP-URLAUTH]	Section 5.7
URL-PARTIAL	Section 5.7.1	Section 5.7
$Forwarded keyword	-	Section 5.9
$SubmitPending keyword	-	Section 5.10
$Submitted keyword	-	Section 5.10

In addition to this list, any Lemonade Message Stores MUST send the CAPABILITY response code (see Section 7.1 of [IMAP]) in the initial server greeting and after the LOGIN/AUTHENTICATE commands.

3.3. Lemonade Message Delivery Agents

Lemonade Message Delivery Agents MUST support Sieve mail filtering language as described in [SIEVE], and MUST support the following Sieve extensions. Note that the Lemonade Profile imposes further requirements for some cases, detailed in the sections cited.

Sieve extension	Reference	Requirements
ENOTIFY	[SIEVE-NOTIFY]	Section 6
IMAP4FLAGS	[SIEVE-IMAP4FLAGS]	Section 6
RELATIONAL	[SIEVE-RELATIONAL]	Section 6
VACATION	[SIEVE-VACATION]	Section 6
VARIABLES	[SIEVE-VARIABLES]	Section 6
comparator-i;unicode-casemap	[UNICODE-CASEMAP]	Section 6

Lemonade Message Delivery Agents should also consider supporting a Sieve script management protocol, such as [MANAGESIEVE].

4. Lemonade Submission Servers

All Lemonade Submission Servers implement the Mail Submission protocol described in [SUBMIT], which is in turn a specific profile of [ESMTP]. Therefore any MUA designed to submit email via [SUBMIT] or [ESMTP] will interoperate with Lemonade Submission Servers.

In addition, Lemonade Submission Servers implement the following set of SMTP and Submission extensions to increase message submission efficiency.

4.1. Forward Without Download

In order to optimize network usage for the typical case where message content is copied to, or sourced from, the IMAP store, Lemonade provides support for a suite of extensions collectively known as Forward Without Download, discussed in detail in Section 8.

Lemonade Submission Servers MUST support BURL [SMTP-BURL], 8BITMIME [SMTP-8BITMIME], BINARYMIME [SMTP-BINARYMIME] and CHUNKING [SMTP-BINARYMIME] SMTP extensions.

BURL MUST support URLAUTH type URLs [IMAP-URLAUTH], and thus MUST advertise the "imap" option following the BURL EHLO keyword (See [SMTP-BURL] for more details).

4.2. Pipelining

Some clients regularly use networks with a relatively high latency, such as Mobile or Satellite based networks. Avoidance of round-trips within a transaction has a great advantage for the 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 [SMTP-PIPELINING].

4.3. DSN Support

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

4.4. Message size declaration

There is a distinct advantage in detecting failure cases as early as possible in many cases, such as where the user is charged per-octet, or where bandwidth is low. This is especially true of large message sizes.

Lemonade Submission Servers MUST support the SMTP Service Extension

for Message Size Declaration [SMTP-SIZE].

Lemonade Submission Servers MUST expand all BURL parts before evaluating if the supplied message size is acceptable.

A Lemonade capable client SHOULD use message size declaration. In particular the client MUST NOT send a message to a mail submission server, if it knows that the message exceeds the maximal message size advertised by the submission server. When including a message size in the MAIL FROM command, the client MUST use a value that is at least as large as the size of the assembled message data after resolution of all BURL parts.

4.5. Enhanced status code Support

Lemonade compliant mail submission servers MUST support SMTP Service Extension for Returning Enhanced Error Codes [SMTP-STATUSCODES]. These allow a client to determine the precise cause of failure.

4.6. Encryption and Compression

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

Support for the DEFLATE compression method, as described in [TLS-COMP], is RECOMMENDED.

5. Lemonade Message Stores

All Lemonade Message Stores implement Internet Message Access Protocol, as defined in [IMAP]. Therefore any MUA written to access messages using the facilities described in [IMAP] will interoperate with a Lemonade Message Store.

In addition, Lemonade Message Stores provide a set of extensions to address the limitations of some clients and networks.

5.1. Quick resynchronization

Resynchronization is a costly part of an IMAP session, and mobile networks are generally more prone to unintended disconnection, which in turn makes this problem more acute. Therefore Lemonade Message Stores provide a suite of extensions to reduce the synchronization cost.

Lemonade Compliant IMAP servers MUST support the CONDSTORE [IMAP-CONDSTORE], the QRESYNC [IMAP-QRESYNC] and the ENABLE [IMAP-ENABLE] extensions. These allow a client to quickly resynchronize any mailbox by asking the server to return all flag changes and expunges that have occurred since a previously recorded state. This can also speed up client reconnect in case the transport layer is cut, whether accidentally or as part of a change in network.

[IMAP-SYNC-HOWTO] details how clients perform efficient mailbox resynchronization.

5.2. Message part handling

The handling of message parts, especially attachments, represents a set of challenges to limited devices, both in terms of the bandwidth used, and the capability of the device.

Lemonade Compliant IMAP servers MUST support the BINARY [IMAP-BINARY] extension. This moves MIME body part decoding operations from the client to the server. The decoded data is equal or less in size than the encoded representation, so this reduces bandwidth effectively.

[IMAP-BINARY] allows for servers to refuse to accept uploaded messages containing binary data, by not accepting the Binary content-transfer-encoding; however Lemonade Compliant IMAP servers SHALL always accept binary encoded MIME messages in APPEND commands for any folder.

[IMAP-CONVERT] MUST also be supported by servers, which allows clients to request conversions between media types, and allows for

scaling images, etc. This provides the ability to view attachments (and sometimes body parts) without the facility to cope with a wide range of media types, or to efficiently view attachments.

5.3. Compression

Lemonade Message Stores SHOULD support the Deflate compression algorithm for TLS, as defined in [TLS-COMP], in order to facilitate compression at as low a level as possible.

However, the working group acknowledges that for many endpoints, this is a rarely deployed technology, and as such, Lemonade Message Stores MUST provide [IMAP-COMPRESS] support for fallback application-level stream compression, where TLS is not actively providing compression.

5.4. Notifications

The addition of Server-to-client notifications transforms the LEMONADE profile into an event-based synchronization protocol. Whenever an event occurs that interests the MUA, a notification can be generated. The LEMONADE WG used the notifications architecture shown in [LEMONADE-NOTIFICATIONS] to develop the Lemonade Profile.

If the MUA is connected to the IMAP server, inband notifications are generated using the facilities outlined in Section 5.4.1.

When the MUA is not connected, the Notification filter generates a outband notification. The notification filter may be considered as acting on a push email repository.

If the MUA is not connected, and outband notification is disabled, the client must perform a quick-sync on reconnect to determine mailbox changes, using the mechanisms outlined in Section 5.1.

5.4.1. IMAP Notifications

Lemonade Message Stores MUST support the IDLE [IMAP-IDLE] extension. The extension allows clients to receive unsolicited notifications about changes in the selected mailbox, without needing to poll for changes. The responses forming these notifications MUST be sent in a timely manner when such changes happen.

Lemonade Message Stores also provide the NOTIFY extension described in [IMAP-NOTIFY], which allows clients to request specific event types to be sent immediately to the client, both for the currently selected folder and others. Such event types include message delivery, and mailbox renames.

5.4.2. External Notifications

Lemonade, and TCP, provide for long-lived idle connections between the client and mail store, allowing the server to push notifications within IMAP. Some mobile networks support dormancy, which shuts down the radio traffic channel during idle periods to conserve handset and network resources, while maintaining IP and TCP state. (See the [LEMONADE-DEPLOYMENTS] document for more information.)

However, there are environments where the email client cannot remain active indefinitely, or where it is not advisable (or even always possible) for TCP connections to the server to remain up while idle for extended periods. In these situations, a good user experience requires that when "interesting" events occur in the mail store, the client be informed so that it can connect and resynchronize. At an absolute minimum, this requires that at least the arrival of new mail generate some sort of wake-up to the email client. A number of vendors have implemented various solutions to this. As examples of what has been done, for many years (long pre-dating cellular handsets) the technique described in [FINGER-HACK] has been supported. Today, a number of email vendors include facilities to send SMS or other simple non-stream messages to clients on handsets when new mail arrives. OMA has published a mechanism that uses WAP PUSH to send a basic message containing a URL [OMA-EMN]. The IETF is investigating ways to standardize enhanced functionality in this area.

A "push email" user experience can be achieved using any number of techniques, ranging from always-on TCP connectivity to the server and the NOTIFY extension described above, to OMA EMN, or even a non-standard trigger message over SMS. In any technique, the client learns of the existence of new mail, and decides to fetch information about it, some part of it, or all of it, and then presents this to the user.

5.5. Searching and View Filters

Lemonade Message Stores MUST support the ESEARCH [IMAP-ESEARCH] extension. The extension allows clients to efficiently find the first or last messages, find a count of matching messages, and obtain a list of matching messages in a considerably more compact representation.

Lemonade Message Stores also provide a mechanism for clients to avoid handling an entire mailbox, instead accessing a view of the mailbox. This technique, common in many desktop clients as a client-side capability, is useful for constrained clients to minimize the quantity of messages and notification data.

Lemonade Message Stores therefore MUST implement the CONTEXT=SEARCH, ESORT and CONTEXT=SORT extensions defined in [IMAP-CONTEXT], as well as the SORT extension defined in [IMAP-SORT].

5.6. Mailbox Handling

Lemonade Message Stores MUST support the NAMESPACE [IMAP-NAMESPACE] extension. The extension allows clients to discover shared mailboxes and mailboxes belonging to other users, and provide a normalized hierarchy view of the mailboxes available.

Lemonade Message Stores MUST support the I18NLEVEL=<n> [IMAP-I18N] extension, with <n> having value 1 or 2. It adds support for non-English (internationalized) search and sort functions. (Note that I18NLEVEL=2 implies support for I18NLEVEL=1, so a Lemonade compliant client that make use of this extension MUST recognize either one.)

5.7. Forward Without Download

In order to optimize network usage for the typical case where message content is copied to, or sourced from, the IMAP store, Lemonade provides support for a suite of extensions collectively known as Forward Without Download, discussed in detail in Section 8.

Lemonade Message Stores MUST support CATENATE [IMAP-CATENATE], UIDPLUS [IMAP-UIDPLUS] and URLAUTH [IMAP-URLAUTH]. Lemonade Message Stores MUST also support URL-PARTIAL as described in Section 5.7.1.

5.7.1. Support for PARTIAL in CATENATE and URLAUTH

[IMAP-URL] introduced a new syntactic element for referencing a byte range of a message/body part. This is done using the ;PARTIAL= field. If an IMAP server supports PARTIAL in IMAP URL used in CATENATE and URLAUTH extensions, then it MUST advertise the URL-PARTIAL capability in the CAPABILITY response/response code.

5.8. Additional IMAP extensions

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

Lemonade Message Stores MUST also implement the SASL-IR [IMAP-SASL-IR] extension, which allows clients to save a round trip during authentication, potentially pipelining the entire authentication sequence.

Lemonade Compliant IMAP servers MUST support IMAP over TLS [IMAP] as

required by [IMAP]. As noted above in Section 5.3, servers SHOULD support the deflate compression algorithm for TLS, as specified in [TLS-COMP].

5.9. Registration of $Forwarded IMAP keyword

The $Forwarded IMAP keyword is used by several IMAP clients to specify that the marked message was forwarded 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 Message Stores MUST be able to store the $Forwarded keyword. They MUST preserve it on the COPY operation. The servers MUST support the SEARCH KEYWORD $Forwarded.

5.10. Registration of $SubmitPending and $Submitted IMAP keywords

The $SubmitPending IMAP keyword designates the message as awaiting to be submitted. This keyword allows to store messages waiting to be submitted in the same mailbox where messages that were already submitted and/or being edited are stored. A mail client sets this keyword when it decides that the message needs to be sent out. When a client (it might be a different client from the one that decided that the message is pending submission) starts sending the message, it atomically (using "STORE (UNCHANGEDSINCE)") adds the $Submitted keyword. Once submission is successful, the $SubmitPending keyword is atomically cleared. The two keywords allow to distinguish messages being actively submitted (messages that have both $Submitted and $SubmitPending keywords set) from messages awaiting to be submitted, or from messages already submitted. They also allow to find all messages that were supposed to be submitted, if the client submitting them crashes or quits before submitting them.

Lemonade Message Stores MUST be able to store the $SubmitPending and the $Submitted keyword. Lemonade Message Stores MUST preserve them on the COPY operation. The servers MUST support the SEARCH KEYWORD $SubmitPending and SEARCH KEYWORD $Submitted.

5.11. Related IMAP extensions

Section 5.11 is non-normative.

Server implementations targeting to fulfil OMA MEM requirements [OMA-MEM-REQ] should consider implementing [IMAP-FILTERS], which provides a way to persist definition of virtual mailboxes on the

server. They should also consider implementing METADATA-SERVER [METADATA] extension, which provides a way of storing user-defined data associated with a user account.

6. Lemonade Message Delivery Agents

Lemonade Message Delivery Agents MUST support the [SIEVE] filtering language at the point of delivery, allowing the user to control which messages are accepted, and where they are filed.

Lemonade Message Delivery Agents MUST support the Sieve Vacation extension [SIEVE-VACATION], which allows the client to setup an auto-responder, typically to report being on vacation (thus the name of the Sieve extension).

Lemonade Message Delivery Agents MUST support the Sieve Enotify extension [SIEVE-NOTIFY], which allows a Sieve script to generate notifications (such as XMPP, SIP or email) about received messages.

Lemonade Message Delivery Agents MUST support the Sieve Variables extension [SIEVE-VARIABLES], which adds support for variables to the Sieve scripting language. This extension is typically used with Sieve Enotify or Vacation to customize responses/notifications.

Lemonade Message Delivery Agents MUST support the Sieve Relational extension [SIEVE-RELATIONAL], which adds support for relational comparisons to the Sieve scripting language. This extension is typically used together with Sieve Enotify.

Lemonade Message Delivery Agents MUST support the Sieve Imap4Flags extension [SIEVE-IMAP4FLAGS], which allows a Sieve script to set IMAP flags/keywords when delivering a message to a mailbox. For example this can be used to automatically mark certain messages as interesting, urgent, etc.

Lemonade Message Delivery Agents MUST support the i;unicode-casemap comparator in Sieve [UNICODE-CASEMAP], which is declared as "comparator-i;unicode-casemap" in the Sieve "require" statement. The comparator allows for case-insensitive matching of Unicode characters.

Lemonade Message Delivery Agents should consider supporting Sieve script management using the [MANAGESIEVE] protocol. If they do, they MUST also advertise in [MANAGESIEVE] all Sieve extensions listed in this section.

7. Lemonade Mail User Agents

Although all existing IMAP MUAs are Lemonade compliant in as much as all Lemonade services are based on the existing [IMAP] and [SUBMIT] protocols, client implementors are encouraged to take full advantage of the facilities provided by Lemonade Submission Servers and Lemonade Message Stores, as described in Section 4 and Section 5 respectively.

Note that the explicit usage of [SUBMIT] means that when opening a connection to the submission server, clients MUST do so using port 587 unless explicitly configured to use an alternate port [RFC5068]. If the TCP connection to the submission server fails to open using port 587, the client MAY then immediately retry using a different port, such as 25. See [SUBMIT] information on why using port 25 is likely to fail depending on the current location of the client, and may result in a failure code during the SMTP transaction.

In addition, some specifications are useful to support interoperable messaging with an enhanced user experience.

Lemonade capable clients SHOULD support the Format and DelSp parameters to the text/plain media type described in [FLOWED], and generate this format for messages.

Lemonade capable clients SHOULD support, and use, the $Forwarded keyword described in Section 5.9.

8. Forward without download

8.1. Motivations

The advent of client/server email using the [IMAP] and [SUBMIT] protocols changed what formerly were local disk operations to become repetitive network data transmissions.

Lemonade "forward without download" makes use of the [SMTP-BURL] 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] 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 POP, NNTP, or whatever else may be designed in the future.

8.2. 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 MUA. Frequently, users choose to save more complex messages on an [IMAP] 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 [ESMTP] infrastructure, typically using the [SUBMIT] protocol.

8.3. Traditional Strategy

Traditionally, messages are initiated, edited, and assembled entirely within an MUA, although drafts may be saved to an [IMAP] server and later retrieved from the server. The completed text is then transmitted to an 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.

8.4. A new strategy

The model distinguishes between a Messaging User Agent (MUA), an IMAPv4Rev1 Server ([IMAP]) and a SMTP submit server ([SUBMIT]), as illustrated in Figure 1.

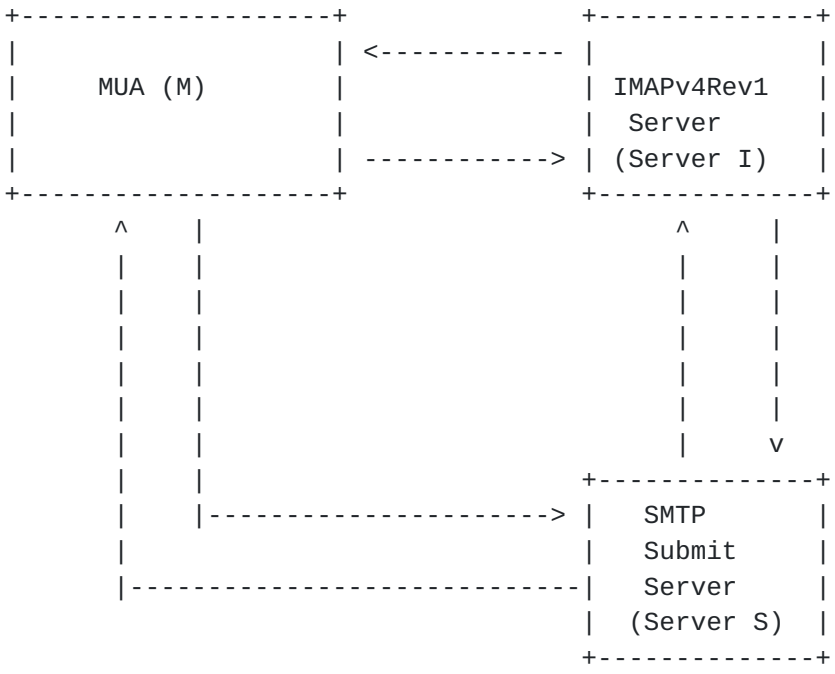

Figure 1: Lemonade "forward without download"

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 extended APPEND command [IMAP-CATENATE] to edit a draft message in the message store and cause the message assembly on the IMAP server. This is most often used when a copy of the message is to be retained on the IMAP server, as discussed in Section 8.6.

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.

8.4.1. Message assembly using IMAP CATENATE extension

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

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

```
      M: A0053 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: A0053 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 [IMAP-URLAUTH] 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 8.4.2 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=
M: MAIL FROM:<bob.ar@example.org>
M: RCPT TO:<foo@example.net>
S: 235 2.7.0 PLAIN authentication successful.
S: 250 2.5.0 Address Ok.
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 [IMAP-URLAUTH] for details of the semantics and steps. The so-called "pawn-ticket" authorization mechanism uses a URI which 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 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: A0054 UID STORE 25627 +FLAGS.SILENT ($Forwarded)
I: * 215 FETCH (UID 25627 MODSEQ (12121231000))
I: A0054 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, as it simply changes non-operational metadata not essential to client operations or interoperability. The untagged FETCH response is due to [IMAP-CONDSTORE]. The $Forwarded IMAP keyword is described in

Section 5.9.

8.4.2. Message assembly using SMTP CHUNKING and BURL extensions

In the [IMAP-URLAUTH]/[SMTP-BURL] variant of the Lemonade "forward without download" strategy, messages are initially composed and edited within an MUA. During submission [SUBMIT], BURL [SMTP-BURL] and BDAT [SMTP-BINARYMIME] 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 [IMAP-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 [IMAP]).

Example:

```
M: B0051 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: B0051 OK completed
```

M: {to I} The client uses GENURLAUTH command to request URLAUTH URLs (See [IMAP-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 [IMAP-URLAUTH] for details of the semantics and steps).

```
M: B0052 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: B0052 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 [SMTP-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 8.4.1 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 [SMTP-BURL] for details of the semantics and steps. The so-called "pawn-ticket" authorization mechanism uses a URI which 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: b000 STARTTLS
      I: b000 Start TLS negotiation now
      ...TLS negotiation, if successful - subsequent data
          is encrypted...
      S: b001 LOGIN submitserver secret
      I: b001 OK submitserver logged in
      S: b002 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: b002 OK URLFETCH completed
      I: b003 LOGOUT
      S: * BYE See you later
      S: b003 OK Logout successful
```

Note that 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 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: B0053 UID STORE 25627 +FLAGS.SILENT ($Forwarded)
I: * 215 FETCH (UID 25627 MODSEQ (12121231000))
I: B0053 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. As in the previous example, this command is not critical, and can be omitted. The untagged FETCH response is due to [IMAP-CONDSTORE]. The $Forwarded IMAP keyword is described in Section 5.9.

8.5. Security Considerations for pawn-tickets.

The so-called "pawn-ticket" authorization mechanism uses a URI, which contains its own authorization credentials using [IMAP-URLAUTH]. The advantage of this mechanism is that the SMTP submit [SUBMIT] server cannot access any data on the [IMAP-URLAUTH] 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.

8.6. Copies of Sent messages: The fcc problem

The "fcc problem" refers to delivering a copy of a message to a mailbox, or "file carbon copy". 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" which represents the

"problem" aspect of the "fcc problem".

The BURL [SMTP-BURL] extension can be used to eliminate the additional transmission. The final message is uploaded to the mailbox designed for outgoing mail, by the APPEND command of [IMAP]. Note that when doing so the client ought to use the $SubmitPending and $Submitted IMAP keywords described in Section 5.10. Also note that APPEND, including when enhanced by [IMAP-CATENATE], can only create a single copy of the message and this is only of use on the server which stages the outgoing message for submission. Additional copies of the message on the same server can be created by using one or more COPY commands.

9. Deployment Considerations

Deployment considerations are discussed extensively in [LEMONADE-DEPLOYMENTS].

10. Security Considerations

Implementors are advised to examine the Security Considerations of all the referenced documents. This section merely highlights these, and advises implementors on specific issues relating to the combination of extensions.

Security considerations on Lemonade "forward without download" are discussed throughout Section 8. Additional security considerations can be found in [IMAP] 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 [SMTP-AUTH]. The mandatory-to-implement authentication mechanism for IMAP is described in [IMAP].

10.1. 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 is 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.

10.2. TLS

When Lemonade clients use the BURL extension for mail submission, an extension that 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 both the IMAP session that issues GENURLAUTH and the mail submission path will provide protection against this attack.

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 which 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.

10.3. Additional extensions and deployment models

This specification provides no additional security measures beyond those in the referenced Internet Mail and Lemonade documents.

We note however the security risks associated with:

- Outband notifications
- Server configuration by client
- Client configuration by server
- Presence of proxy servers
- Presence of servers as intermediaries
- In general the deployment models considered by OMA MEM that are not conventional IETF deployment models.
- Measures to address a perceived need to traverse firewalls and mobile network intermediaries.

Deployments that provide these additional services or operate in these environments need to consult the security considerations for the relevant standards and organizational security practices.

11. IANA considerations

IMAP4 capabilities are registered by publishing a standards track or IESG approved experimental RFC. The registry is currently located at <http://www.iana.org/assignments/imap4-capabilities>. This document defines the URL-PARTIAL IMAP capability Section 5.7.1. IANA is requested to add this extension to the IANA IMAP Capability registry.

12. Version history

- Version 11:
  - Made ManageSieve reference Informative.
  - Renamed $PendingSubmission to $SubmitPending, dropped $BeingSubmitted and added $Submitted IMAP keywords.
- Version 10:
  - Added ManageSieve.
  - Added a requirement to support i;unicode-casemap in Sieve (already required in IMAP).
  - Added description of various Sieve extensions.
  - Added definition of URL-PARTIAL IMAP extension.
  - Added definition of $PendingSubmission and $BeingSubmitted IMAP keywords.
- Version 9:
  - Updated references and removed extensions that there was an agreement to remove.
- Version 08:
  - Removed LIST-EXTENDED, WITHIN and QUICKSTART.
  - Updated names of required extensions when they were renamed (e.g. COMPARATOR ==> I18NLEVEL=1).
- Version 06:
  - Updated references and added extensions agreed at the Lemonade interim in Spring 2007.
  - Require any Lemonade compliant IMAP server to support CAPABILITY response code.
- Version 04:
  - Major reorganization of text.
  - Move checklist summary to beginning of document, collate Submission and Store server requirements.
- Version 03:
  - Replaced RECONNECT (server side quick reconned) with QRESYNC (client side quick reconnect)
  - Added WITHIN and LIST-EXTENDED.
  - Moved IDLE extension to a separate section.
  - Added requirement for clients to use Format=flowed.
- Version 02:
  - Update of references and how they are displayed in the text (Comments from Randy Gellens)
  - Update of list of extensions to support as MUST by the Lemonade Profile Bis
  - Update of options for compression via placeholder imap-compression section describing compression requirements
  - Update of support of TCP chalenged environments
  - Update of support of object level encryption
  - Clarified the use of $Forwarded in the examples, and demonstrated how to remove the \Draft flag from the sent message

    * Clarified $Forwarded
    * Added RECONNECT to imap-condstore section
    * Add new section imap-bodypart, "Message part handling", describing BINARY and CONVERT requirements
    * Added placeholder section for notifications
    * Added various extensions to imap-other section, and added clarifying comments to IDLE, NAMESPACE, and a further references to TLS DEFLATE compression
    * Added extension names to IMAP table
    * Fixed all issues found with original Lemonade profile so far.
- Version 01:
    * Lemonade profile has been introduced in-line, with some updates / corrections.
    * Subsequent re-organization of the text
    * Details of extensions proper to Lemonade Profile-bis have been updated to refer to the drafts newly accepted as WG IETF drafts.
    * Addition of appendix on attachements streaming.
- Version 00:
    * It evolved from a combination of the content of Lemonade profile and the OMA MEM realization internet draft.

13. Acknowledgements

The editors acknowledge and appreciate the work and comments of the IETF Lemonade working group and the OMA MEM working group.

In particular, the editors would like to thank Eric Burger, Glenn Parsons, Randall Gellens, Filip Navara, Zoltan Ordogh, Greg Vaudreuil, and Fan Xiaohui for their comments and reviews.

14. References

14.1. Normative References

[FLOWED] Gellens, R., "The Text/Plain Format and DelSp Parameters", RFC 3676, February 2004.

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

[IMAP-BINARY] Nerenberg, L., "IMAP4 Binary Content Extension", RFC 3516, April 2003.

[IMAP-CATENATE]

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

[IMAP-COMPRESS]
Gulbrandsen, A., "The IMAP COMPRESS Extension", RFC 4978, August 2007.

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

[IMAP-CONTEXT]
Cridland, D. and C. King, "Contexts for IMAP4", RFC 5267, July 2008.

[IMAP-CONVERT]
Melnikov, A. and P. Coates, "Internet Message Access Protocol - CONVERT Extension", RFC 5259, July 2008.

[IMAP-ENABLE]
Gulbrandsen, A. and A. Melnikov, "The IMAP ENABLE Extension", RFC 5161, March 2008.

[IMAP-ESEARCH]
Melnikov, A. and D. Cridland, "IMAP4 Extension to SEARCH Command for Controlling What Kind of Information Is Returned", RFC 4731, November 2006.

[IMAP-I18N]
Newman, C., Gulbrandsen, A., and A. Melnikov, "Internet Message Access Protocol Internationalization", RFC 5255, June 2008.

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

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

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

[IMAP-NOTIFY]
Gulbrandsen, A., King, C., and A. Melnikov, "The IMAP NOTIFY Extension", RFC 5465, February 2009.

[IMAP-QRESYNC]
Melnikov, A., Cridland, D., and C. Wilson, "IMAP4 Extensions for Quick Mailbox Resynchronization", draft-ietf-lemonade-5162bis-00 (work in progress), October 2008.

[IMAP-SASL-IR]
Siemborski, R. and A. Gulbrandsen, "IMAP Extension for Simple Authentication and Security Layer (SASL) Initial Client Response", RFC 4959, September 2007.

[IMAP-SORT]
Crispin, M. and K. Murchison, "Internet Message Access Protocol - SORT and THREAD Extensions", RFC 5256, June 2008.

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

[IMAP-URL]
Melnikov, A. and C. Newman, "IMAP URL Scheme", RFC 5092, November 2007.

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

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

[SIEVE] Guenther, P. and T. Showalter, "Sieve: An Email Filtering Language", RFC 5228, January 2008.

[SIEVE-IMAP4FLAGS]
Melnikov, A., "Sieve Email Filtering: Imap4flags Extension", RFC 5232, January 2008.

[SIEVE-NOTIFY]
Melnikov, A., Leiba, B., Segmuller, W., and T. Martin, "Sieve Email Filtering: Extension for Notifications", RFC 5435, January 2009.

[SIEVE-RELATIONAL]
Segmuller, W. and B. Leiba, "Sieve Email Filtering: Relational Extension", RFC 5231, January 2008.

[SIEVE-VACATION]
Showalter, T. and N. Freed, "Sieve Email Filtering: Vacation Extension", RFC 5230, January 2008.

[SIEVE-VARIABLES]
Homme, K., "Sieve Email Filtering: Variables Extension", RFC 5229, January 2008.

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

[SMTP-AUTH]
Siemborski, R. and A. Melnikov, "SMTP Service Extension for Authentication", RFC 4954, July 2007.

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

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

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

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

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

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

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

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

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[TLS-COMP] Hollenbeck, S., "Transport Layer Security Protocol Compression Methods", RFC 3749, May 2004.

[UNICODE-CASEMAP] Crispin, M., "i;unicode-casemap - Simple Unicode Collation Algorithm", RFC 5051, October 2007.

14.2. Informative References

[ESMTP] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, October 2008.

[FINGER-HACK] Gellens, R., "Simple New Mail Notification", RFC 4146, August 2005.

[IMAP-FILTERS] Melnikov, A. and C. King, "IMAP4 Extension for Named Searches (Filters)", RFC 5466, February 2009.

[IMAP-SYNC-HOWTO] Melnikov, A., "Synchronization Operations for Disconnected IMAP4 Clients", RFC 4549, June 2006.

[LEMONADE-ARCH] Burger, E. and G. Parsons, "LEMONADE Architecture - Supporting Open Mobile Alliance (OMA) Mobile Email (MEM) using Internet Mail", draft-ietf-lemonade-architecture-04 (work in progress), November 2008.

[LEMONADE-DEPLOYMENTS] Gellens, R., "Deployment Considerations for Lemonade-Compliant Mobile Email", BCP 143, RFC 5383, October 2008.

[LEMONADE-NOTIFICATIONS] Gellens, R. and S. Maes, "Lemonade Notifications Architecture", draft-ietf-lemonade-notifications-10 (work in progress), July 2008.

[MANAGESIEVE] Melnikov, A. and T. Martin, "A Protocol for Remotely

Managing Sieve Scripts", draft-martin-managesieve-12 (work in progress), September 2008.

[METADATA] Daboo, C., "The IMAP METADATA Extension", RFC 5464, February 2009.

[OMA-EMN] Open Mobile Alliance, "Open Mobile Alliance Email Notification Version 1.0", OMA http://www.openmobilealliance.org/Technical/release_program/emn_v10.aspx, October 2007.

[OMA-MEM-ARCH] Open Mobile Alliance, "Mobile Email Architecture Document", OMA (Work in Progress), http://www.openmobilealliance.org/, October 2005.

[OMA-MEM-REQ] Open Mobile Alliance, "Mobile Email Requirements Document", OMA http://www.openmobilealliance.org/release_program/docs/RD/OMA-RD-MobileEmail-V1_0_20051018-C.pdf, Oct 2005.

[RFC5068] Hutzler, C., Crocker, D., Resnick, P., Allman, E., and T. Finch, "Email Submission Operations: Access and Accountability Requirements", BCP 134, RFC 5068, November 2007.

Authors' Addresses

Dave Cridland (editor)
Isode Limited
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: dave.cridland@isode.com

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

Email: Alexey.Melnikov@isode.com

Stephane H. Maes (editor)
Oracle
MS 4op634, 500 Oracle Parkway
Redwood Shores, CA 94539
USA

Phone: +1-203-300-7786
Email: stephane.maes@oracle.com