Internet Draft: Lemonade Notifications and Filters

Document: draft-ietf-lemonade-notifications-04.txt
Expires: October 2007

S. H. Maes
R. Cromwell
Oracle

R. Gellens
Qualcomm

April, 2007

Lemonade Notifications and Filters

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The IETF Trust (2007). All Rights Reserved.

Abstract

This document discusses how to provide notification and filtering mechanisms to IMAP as part the Lemonade profile.

This document also discusses the use of Lemonade notifications to implement server to server notifications.

Table of Contents

1. Conventions Used in this Document

In examples, "C:" and "S:" indicate lines sent by the client and 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].

An implementation is not compliant if it fails to satisfy one or more of the MUST or REQUIRED level requirements for the protocol(s) it implements. An implementation that satisfies all the MUST or REQUIRED level and all the SHOULD level requirements for a protocol is said to be "unconditionally compliant" to that protocol; one that satisfies all the MUST level requirements but not all the SHOULD level requirements is said to be "conditionally compliant." When describing the general syntax, some definitions are omitted as they are defined in [RFC3501].

2. Introduction

As the work on LEMONADE Profile ([LEMONADEPROFILE] and [LEMONADEPROFILEBIS]) progresses, a need has been identified to provide notification and filtering mechanisms to IMAP4.

The requirements for inband and outband server to client notifications are documented in [OMA-ME-RD].

3. Objectives

According to these analyses, there is a need to support:
Mechanisms for event-based (server to client) synchronization:
* Defines the relationship between notification mechanisms and the IMAP4 protocol
- To minimize the latency observed for email events on the email server to be reflected in the email client.
- To avoid unnecessary polling and requests from the e-mail clients:
. To reduce the total amount of data to be exchanged between email server and client, e.g. by allowing the email client to select which messages to synchronize and how to synchronize.
. To reduce the amount of transactions.
* Needs to cope with possible lost or delayed notifications

* Support in-band (within IMAP band) and out-band notifications (Exchanged via other servers / enablers).
- Specified in ways that are network transport independent but may contain some bindings to particular notification channels (e.g. SMS binary, WAP Push, SIP Notification, ...)
- When the email client is connected to the email server,

only inband notifications is expected take place
* Defines notification payload for inband and outband mechanisms.
# Server-side filtering to decide which messages will be accessible by the email client.
* Filtering results into the following logical types:
- Type A: Messages filtered out and not accessible by the email client (no notification, no header access, no access)
- Type B: Messages that are accessible by the mobile e-mail enabler client but no outband notification takes place. Inband notification might however take place if email client is already connected to email server.
- Type C: Messages that are accessible by the e-mail client for which notifications (outband or inband) are always sent to the email client.
Notions of Filters:
* View filters: Filters that determine which email messages are of type B and C or A
* Notification filters: Filters that determine which email messages are of type C or B
* Event filters: Filters that determines what events are to be notified to the client
# Mechanisms to allow the user to update the filters from the email client
# Mechanisms to allow configuration and exchange of settings between the client and the server in band or outband: - Server to client: e.g. server ID, account name, policies, - Client to server: e.g. rules filters vacation notices, notification channel, ...

The present document describes how this may be achieved within the scope of LEMONADE Profile [LEMONADEPROFILEBIS].

4. Notification logical architecture and LEMONADE Profile bis

The target logical architectures involving the LEMONADE Profile and notifications are discussed in [LEMONADEPROFILEBIS].

Figure 1 illustrates how notification and filtering can be introduced in the context of LEMONADE profile bis.

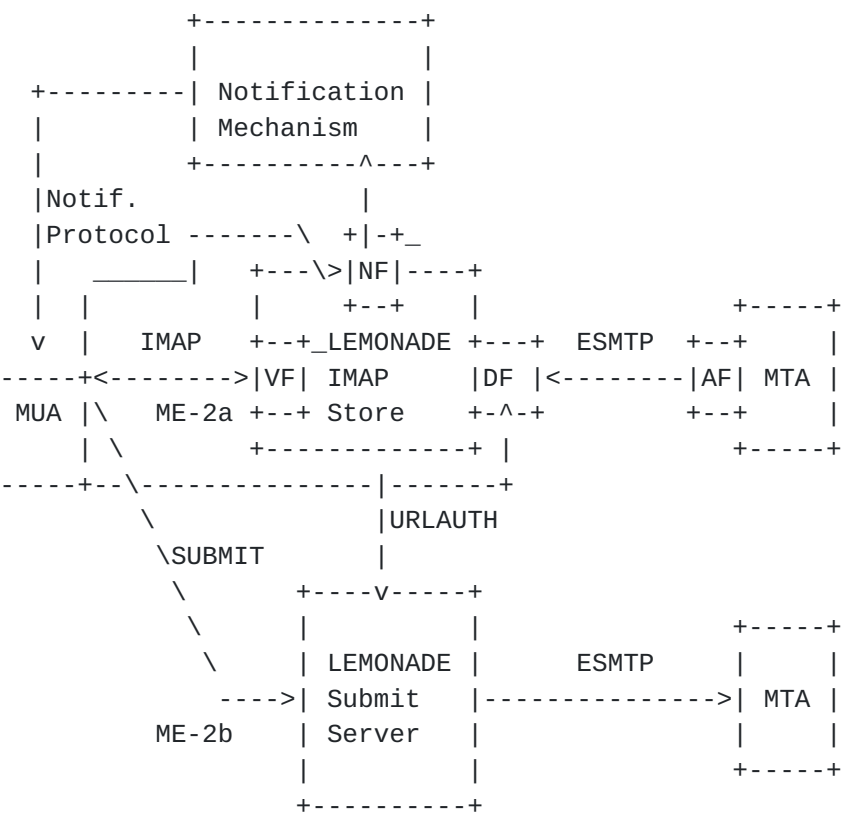

Figure 1: Filtering mechanism defined in LEMONADE Profile bis architecture.

In Figure 1, four categories of filters are defined:

AF: Administrative Filters - Set up by email service provider. AF are typically not configured by the user and set to apply policies content filtering, virus protection, spam filtering etc...

DF: Deposit Filters - Filters that are executed on deposit of new emails. They can be defined as SIEVE filters [SIEVE]. They can include vacation notices.
VF: View Filters - Filters that define which emails are visible to the MUA. View filters can be defined as virtual folders [VFOLDER] as described in later in this document.
NF: Notification Filters - Filters that define for what email server event an outband notification is sent to the client.

The filters are manageable from the MUA:

NF and DF: via SIEVE management protocol <Editor's note: Still to be defined>. [IMAPSIEVE] provides an example of how notification filters (NF) may be expressed in SIEVE.
VF: via VFOLDER as defined in [VFOLDER]

5. Capability

A server supporting Lemonade notifications MUST report the following set of capabilities: IDLE, METADATA, LISTEXT, LNOTIFICATION, VFOLDER.

METADATA (and by transitive dependency LISTEXT) are from the [ANNOTATEMORE] extension, used to store notification provisioning and preference information.

VFOLDER declares support for [VFOLDER].

LNOTIFICATION is currently a placeholder umbrella capability declares support for outband notification filters and filter management as described in this document, which may includes works in progress such as SIEVE notification filters and filter management.

6. Event-based synchronization

The addition of Server-to-client notifications transforms the LEMONADE profile into an event-based synchronization protocol. Whenever an event of the type [MSGEVENTS] occurs within the view, a notification can be generated. [MSGEVENTS] provides a list of

possible events that could be notified (based on the filter settings).

If the MUA is connected to the IMAP server, inband notifications are generated following IDLE [RFC2177].

When the MUA is not connected, the Notification filter generates a outband notification. The notification filter may be considered as acting on a PUSH 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. If the MUA has only momentarily lost connection, it should also attempt to use [RECONNECT].

Formally, a repository consists of a set of folders, and each folder has both a name and a set of messages associated with it. The "complete repository" consists of all folders of an end user and all the associated messages for each of those folders.

This is illustrated in Figure 2.

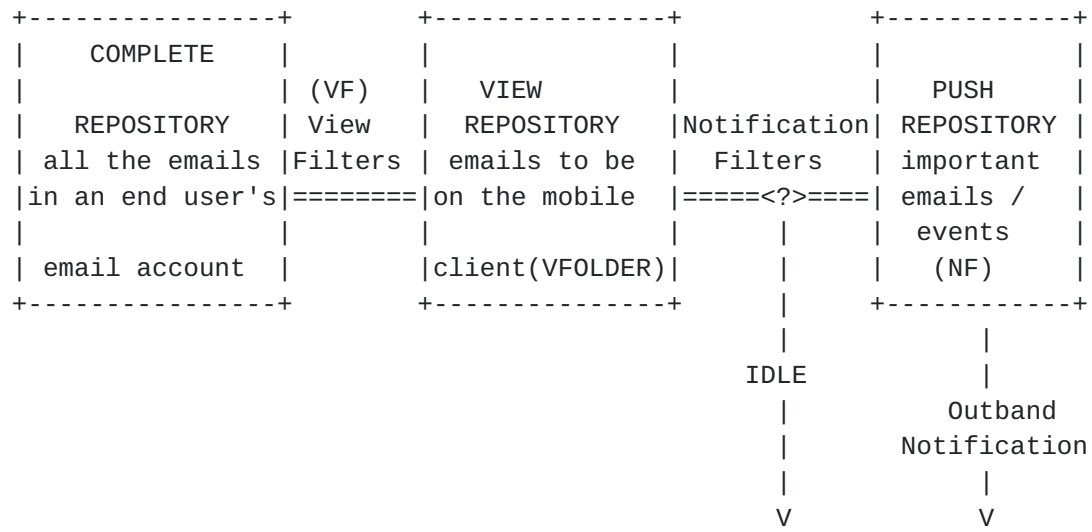

Figure 2: Filters and Repositories

In inband notification mode, the MUA issues IDLE, and notifications are sent as unsolicited responses to the MUA from the LEMONADE IMAP server.

In outband notification mode, the outband notification may be a message (notification payload) and possibly a set of surrounding

exchanges sent with an appropriate format to a particular IP address and port. This may be the address of the MUA. However, in general, it conforms to the interface of a notification server / mechanisms responsible for finalizing the format and sending the notification to the MUA on the client with the appropriate protocol.

The following sections provide description of the outband notification payload format and mechanisms to check, set and notification settings.

7. Filters

[SIEVE] provides an email filtering language.

[VFOLDER] describes how the View Filter (VF) can be implemented and modified from the MUA. Alternative mechanisms for creating virtual mailbox views may be possible, however they should satisfy the requirement that an out-of-band notification may refer to them by name in order to tell the client which view an event occurred for.

[SIEVENOTIFICATION] and [IMAPSIEVE] provides a mechanisms to associate notifications based on the [SIEVE] filter whenever messages are created or changed within a LEMONADE IMAP store.

7.1. Next steps and future work

[SIEVE] filters and [IMAPSIEVE] should be extended to support binding to all [MSGEVENTS].

8. Inband notification payload

Inband notification syntax follows the IDLE specifications [RFC2177]. However, which unsolicited responses are generated by each event is ambiguous in [RFC2177]. [IMAP-EVENTS] defines a more rigorous set of requirements for IDLE events, including how to monitor mailboxes when not in a selected state.

In lieu of this, simpler MUAs MAY choose to interpret unsolicited responses in IDLE as a "wake up call" to perform a sync. [IMAP-DISC] is an informative reference for how to do this

efficiently.

9. Outband Notification payload

9.1. Outband Notification Payload in Clear Text

The outband notification payload is in general in clear text, unless the payload carries sensitive data. Server to Client Notification and Filtering treats the notification as a signal to the client to fetch the information on the server that awaits it.

The payload for wake-up notifications is the OMA EMN format, see [OMA-EN]. This notification, minimally, informs the client that some push event has happened on the server, so it must connect to fetch the information.

Server to client notifications and filters treats the notification as a MUA wake up event to fetch the information on the server that awaits it. The MUA MAY present other behaviors that exploit additional information provided in the notification. However this is out of scope of the specifications.

Wake-up events consists of the following payload (example): <emn mailbox="mailat:john.doe@somewhere.com" timestamp="OMA-EMN-dateformat"> </emn>

The mailbox URI formats are defined in [OMN-EN].

The notification payload is generated by NF and sent to an appropriate messaging interface, appropriately formatted for that interface. The messaging mechanism is then responsible for sending the notification to the MUA.

Example: A new message arrives on the server and this is notified via out-of-band.

```
  S: pushes SMS with the following text:
     <emn
       mailbox="mailat:joe@foo.com"
       timestamp="2004-02-20T06:40:00Z">
     </emn>
```

Here the notification payload is sent to an SMSC messaging interface, appropriately formatted for that interface. The SMSC is responsible for sending the SMS to the MUA using [OMA-EMN] encoding.

If EXTENDED notification format is supported by the MUA, a new extended EMN element is used which contains additional XML attributes.

The format of this extended EMN element is:

```
<!ATTLIST xemn
   mailbox     CDATA    #REQUIRED

   timestamp   CDATA    #IMPLIED

   event       CDATA    #IMPLIED   ;event name from [MSGEVENTS]
   view        CDATA    #IMPLIED   ;IMAP mailbox or view w/message
   sequence-id CDATA    #IMPLIED   ;notification sequence number
   event       CDATA    #IMPLIED   ;event name in [MSGEVENTS]
   uid         CDATA    #IMPLIED   ;message UID
   sender      CDATA    #IMPLIED   ;name of message sender
   datetime    CDATA    #IMPLIED   ;date message was sent
   subject     CDATA    #IMPLIED   ;subject of message
   >
```

With the 'xemn' element's allowed children as

```
<!ELEMENT xemn (#CDATA)>            ;CDATA contains optional snippet
                                     of body of message or any text
                                     to be displayed on MUA when
```



```
                                     viewing message prior to
                                     synchronization.
```

Extended EMN payloads should be encrypted.

Example extended payload:

```
  <xemn
       mailbox="mailat:joe@foo.com"



       view="INBOX"
       event="New Message"
```

```
        timestamp="2004-02-20T06:40:00Z"
        sender="Mike Smith &amp;lt;mike@foo.com&amp;gt;"
        datetime="Thu, 15 May 2006 19:51:21 -0700"
        sequence-id="1"
        uid="157"
        subject="Re: our meeting"
     >
        I can't make the 9:15 meeting, can we reschedule?
    </xemn>
```

A client may treat an extended notification as a wakeup notification, or it may parse the payload and utilize the included data to synchronize new messages more efficiently. Not all of the fields will have values, depending on the type of event. The client may utilize the view and uid attributes to determine which IMAP mailbox the event occurred in, and for which message. The view attribute may refer to an IMAP Mailbox, or a named virtual folder if implemented by the client.

10. LEMONADE message store behavior

Because outband notifications may be sent over unreliable channels (i.e., they may be lost or delayed), the server MUST keep track of the outband notifications that are sent via NF, until the MUA react to them.

If a MUA does not react to outband notifications, the server MAY stop sending outband notifications, except possibly periodic wake up calls until the client reconnects.

11. Provisioning and Preferences for Notification Settings

It is sometimes necessary for the server to inform the client of default notification settings the first time the client is used, as well as notification capabilities of the server. It is also necessary for the client to adjust notification preferences. [ANNOTATEMORE] is used to store provisioning and preference information.

The difference between a provision entry and a preference entry is

that provision entries are typically read-only, global to the user, and represent capabilities, whereas preference entries are writable by the client, and per-mailbox, and represent actual settings.

A client only needs to access provision information once when the device uses the Lemonade server for the first time. It MAY opt to not cache provisioning information, and re-read it on each connection, but it is not necessary. Re-provisioning MAY also be initiated manually via user action, or via out-of-band notification.

Provision and preference data marked "private" in [ANNOTATEMORE] terminology is also inherently per-device, not per-user. Attributes marked "shared" are inherently per-user. If a user logs in as his desktop client identity, "private" preference data is separate from his mobile phone identity, rather than shared. However, "shared" preference data may be visible to all of his devices (but not other users)

11.1. Entry Names and Attributes

Entry names and Attributes listed in this section SHOULD be supported by any server claiming LNOTIFICATION compliance. ABNF structure of attribute values is provided.

11.2. Provision Entries

The following are server-global provision entries.

/notify/outband
Defines attributes related to out-of-band notification.

Attribute Names = Values:
"channel.priv" = "NONE" *(SP ("SMS" / "MMS" / "SIP" / "XMPP" / "UDP") = format)

;a list of supported out-of-band channels
;for the current device and their formats

format = "WAKEUP" / "EXTENDED"

Example: "NONE SMS=WAKEUP MMS=EXTENDED SIP=EXTENDED UDP=EXTENDED"

```
      "credential.priv" = string
                     ;the source address or identity of the server
                      sending the notifications, potentially with an
                      integrity check

   /notify/inband

     Defines attributes related to in-band notifications.

     Attribute Names = Values:
        "push.shared" = event-list
                    ;a space separated list of push event names
                     supported for in-band push. TBD in Lemonade events
                     / notification work.

   /notify/security

     Defines attributes related to negotiable security parameters for
   securing out of band event notification and proxied deployment
   models. TBD.
```

11.3. Preference Entries

The following are per-mailbox per-device preference entries.

```
   /notify/outband
      Defines attributes related to chosen preference for out-of-band
   notification.

   Attribute Names=Values:
      "channel.priv" = "NONE" / "SMS=" format / "MMS=" format /
                       "SIP=" format / "XMPP=" format / "UDP=" format
                        ;for a device capable of multiple mechanisms,
                        ;set this attribute to the desired mechanism


      format = "WAKEUP" / "EXTENDED"


      "address.priv" = string
                       ;the phone number, email address, or
                       ;destination endpoint for notifications
```



```
                       ;if different from server known device default
                       ;(e.g. device phone number, or device IP
```

```
                    ; address on well known open port, etc)


   "limit.priv" = "size=" nz-number  | "total=" nz-number
                    ;the maximum number of bytes to send in
                    ;notification payloads or the total number
                    ;of messages to send. Both per-day

/notify/inband

   Defines attributes related to inband push events.

Attribute Names=Values:
   "msgformat.priv" = fetch-att
                    ;RFC3501 fetch-att syntax detailing the format
                    ;of unsolicited FETCH responses generated in-band
                    ;for new message arrivals

   "push.priv"      = "push" / "pull"
                    ;specifies whether the client expects
                    ;new messages to be pushed via the unsolicited
                    ;FETCH response defined above, or via client
                    ;issued FETCH in response to an EXISTS response.

/notify/event

   Defines attributes related to event / notification filtering.

   "filter.priv"  = "ALL" / "NEW" / "NONE"
                    ;specifies whether no events are pushed
                    ;NEW message events are pushed, or ALL
                    ;events are pushed
```

11.4. Getting and setting preference and provisioning annotations

To fetch a provision, a client should generate a GETANNOTATE command with no mailbox specified according to [ANNOTATEMORE]

Example: Discover all provision parameters

```
  C: a GETMETADATA /notify/* (channel credential push)
  S: * METADATA /notify/outband (channel.priv "SMS=WAKEUP NONE"
                                  credential.priv "55555"
                                  push.shared "new expunged")
  S: a OK GETMETADATA Complete
```

Example: Set out-of-band mechanism to "MMS" with EMN extended support

C: a SETMETADATA "INBOX" /notify/outband (channel.priv "MMS=EXTENDED")
S: a OK SETMETADATA complete

12. Changing filters from the client

VFOLDER also provides ways to update VF.

NF Filter remote management mechanisms MAY rely on [MANAGESIEVE]

13. Out of scope items for IETF

OMA provides ways to perform provisioning via OMA client provisioning and device management. Other provisioning specifications are available (e.g. SMS based).

OMA provides enablers to support outband notifications: the outband notification mechanisms.

Also, OMA XDM may be considered also for outband filter changes.

14. Server to server notifications considerations

The following sections focus on considerations and usage of the Lemonade notifications for server to server notifications

With server to server notifications, a messaging system (e.g. email server, voice mail system, etc.) submits alerts, which describe potential notification events, regarding an end user mailbox status change (e.g. new message has arrived, mailbox is full, etc.).

These alerts are sent to a notification mechanisms, which may, in turn, generate an end user alert notification.

Server to server notifications facilitate a solution where the messaging system initiates an end user notification, while allowing the messaging system, not to be familiar with the end user's notification preferences.

The notification mechanisms are the entities which is familiar with the end user's notification preferences.

Using server to server notifications, allow the messaging system to provide the end user, a unified notification experience (the same look and feel for all messaging systems' accounts), while allowing smooth integration of additional Messaging systems.

14.1. Scope of server to server notifications

The suite of Internet mail protocols (POP3, IMAP4) allows different mail clients to access and manipulate electronic mail messages on a messaging system. These protocols, however, do not provide off-line methods by which an end user can be notified upon changes in the mailbox status. Further, none of the mentioned protocols defines a way to aggregate the status within the end user's various mailboxes.

To provide an end user with the ability of unified Notification and one centralized message-waiting indication (MWI), notification mechanisms are required to aggregate the information of all the events occurring on the end user's different messaging systems.

With server to server notifications, a messaging system notifies the notification mechanisms regarding events occurring in an end user's mailbox. It is important to emphasize, that server to server notifications do not deal with the communications between the notification mechanisms and the end user devices (SMS, WAP Push, MWI, etc.).

For example, when an email message is deposited in an email server, the email server sends a "new message" notification to the notification service, which then notifies the end user by a Short Text Message (SMS).

This process can be extended to include other mailbox events that are important to the end user, such as "mailbox full" and "message rejected" or any other mailbox status change. Each notification should include additional information that is available to the end user such as the mailbox status, message attributes, etc.

The figure 3 depicts the server to server notification scope:

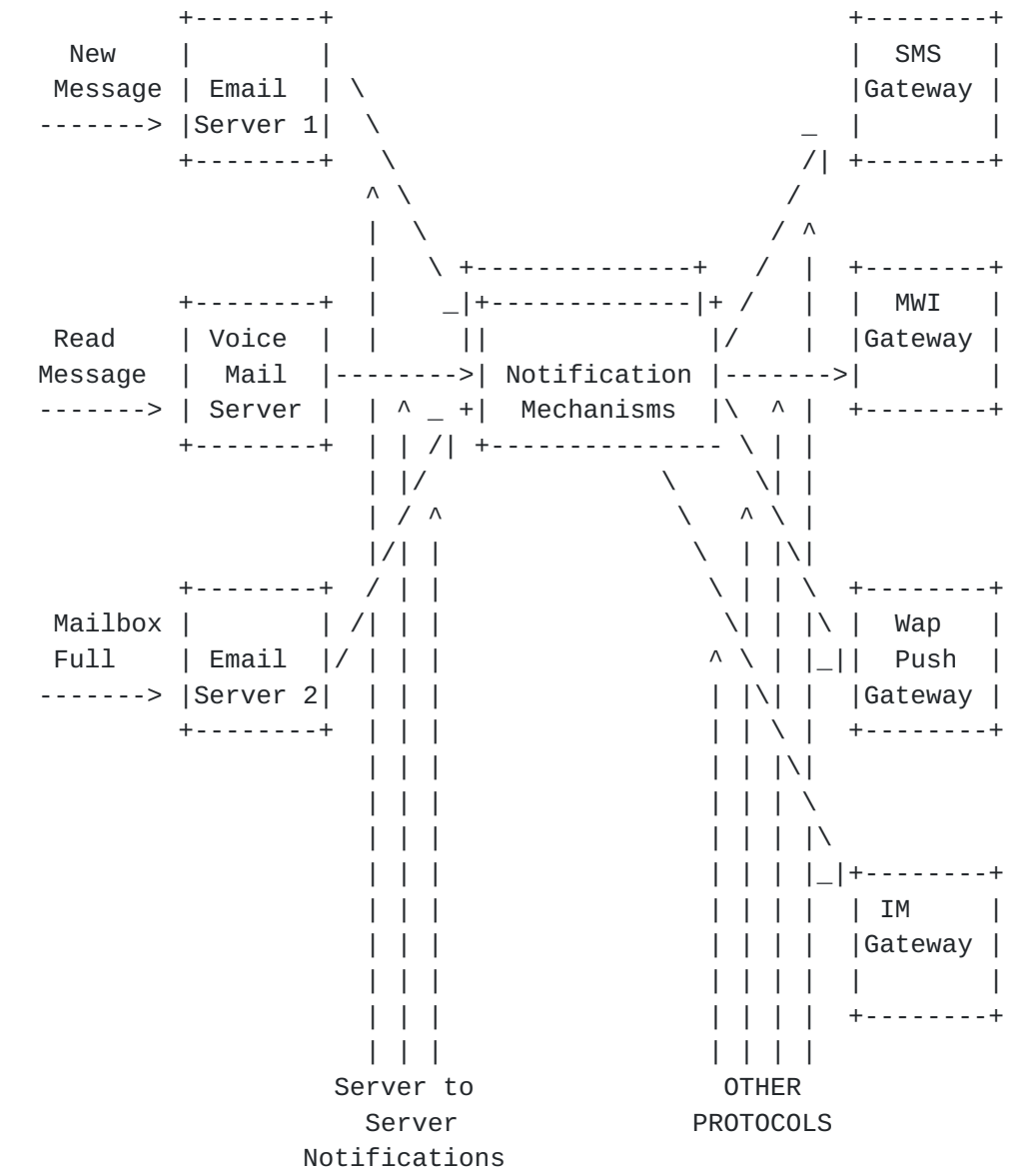

Figure 3: Scope of server to server notifications

The notification mechanisms can either provide an abstract notification mechanism able to convert notifications to an appropriate channel or expose the northbound interface (application interface) exposed by a specific notification channel. In the latter case it may just be a logical concept and the sender of

notifications may directly address the appropriate notification channels. All the options are viable.

14.2. Basic Operation

The messaging system notifies the notification mechanisms (which in turn MAY notify an end user) about events that occurred in the end user's mailbox. Each such notification, referring to a single mailbox event is referred to as a notification request.

The notification request SHOULD contain data regarding the mailbox event which has occurred. It's RECOMMENDED that the request would not contain data regarding the end user notification destinations. This would be left to the notification mechanisms implementation. If such data has been received the notification mechanisms MAY ignore it.

14.3. Server to server terminology

This specification uses the following terms:

Message Waiting Indication (MWI):

A mechanism that indicates to the end user that a message is waiting in a Messaging System

Examples for such action are: SMS message, WAP push message, Instant messaging notification, telephony stutter tone, etc. MWI states may be ON or OFF.

Notification Event:

An event that may result in a notification to the end user or may change the MWI state (ON or OFF)

Messaging System:

A system that maintains a set of one or more mailboxes for end user's messages, for example: email servers, voicemail systems, etc.

Notification Mechanisms:

A system, which aggregates all notification events from multiple Messaging systems to multiple end user destinations.

14.4. Notification payloads

The cases of Figure 1 and Figure 3 are very similar.

In both cases a message must be generated by the message store as result of a message event. This message is communicated to the messaging mechanisms.

Within the context of Lemonade profile (Figure 1), the event is filtered by NF and the payload of the notification is defined in section 8.

In more generic cases, the server to server notification payload can be any message. Certain may be defined to:
Realize the messaging mecanisms task which has caused the notification event. The task may be related to one of the following:
* New message Task
- New Message Deposit
* Mailbox Manipulation Task (e.g. Login, Logout, etc.)
- Login to mailbox
- Logout from mailbox
- Read message
- Delete Message
- Purge Message
* Management Task (e.g. Mailbox Full)
- Mailbox full
- Mailbox full cancellation

The task's types list, as defined above, SHOULD be extendable.
Provide a rich experience to the end user of the notification, without the need to actually retrieve the message. This MAY include mailbox status, message attributes, etc.
Practice different MWI behaviors (e.g. turn MWI indication off after all the messages in all the end user's mailboxes have been read).
URL, as defined in [RFC-URL] or [URLAUTH], referring to the message which has caused the event, to the notification

mecanisms (and eventually, to the end user).

14.5. Server to server notification protocol details

14.5.1. Generic case

Within the more general case of server to server notification, the payload may be an arbitrary text or binary message.

In both cases the interaction model is defined as:

1 An event takes place in the message store
2 The event is filtered. As a result it may be hidden or result into a notification.
3 The notification is a message in a particular payload that is prepared for the target notification mechanisms.
4 The payload is complemented with the necessary information to tell the notification mechanism how and where to send the notification.
5 The complemented payload is then formatted as required by the target notification mechanisms (i.e. the right format on the right port to be sent to the right address, possibly with an appropriate protocol binding e.g. HTTP PUT) plus the information about where / how to send the notification. This last step is imposed by the notification mechanisms and must be known by the notification generating filter.

Different interfaces and bindings may be used depending on the notification channel.

14.5.2. Abstracted notification protocol

When a mechanism is provided to abstractly notify a notification mechanism that is then responsible for notifying via the appropriate channel, the notification protocol MUST follow [NOTIFICATIONPROTOCOL].

14.5.3. Exception Handling

It is assumed that the interface exposed by the notification

mechanisms can notify the messaging system about the outcome of the notification request (notification status message). The notification mechanisms SHOULD notify the messaging system whenever a problem has occurred.

If the request has failed, the response, when available, SHOULD be coherent enough to allow the messaging system to determine the cause of the failure.

The notification mechanisms SHOULD make a distinction between events in which the content of the request has caused an error (request errors), and cases in which a non-source-related reason has caused the error.

The Messaging system SHOULD parse the notification status message, when available, to decide its next actions (e.g. clear the messageËs content, recompile the message data, etc.).

14.6. Server to server complementary information

The server to server complementary information MUST include:

* Identify the end user whose inbox has generated the notification.
* Identify the end user or end target addresses or identifiers that should be informed about the notification event (not necessarily the same as the previous end user).
* Decide what kind of actions, the notification mechanisms should perform, due to the notification request.

14.7. Event orders

For lemonade profile bis, the event order is not important.

For generic server to server notifications, the order may matter and the messaging system must provide the notifications in the order that they are generated by mailbox events.

14.8. Reliability

For Lemonade profile bis, lost or delayed notifications of the MUA

are not critical. A client can recover all missing events next time it connects to the server and the server MUST buffer the notifications and make them available to the MUA when it comes back to the server.

For generic server to server notifications, it is assumed that the data in a notification request is important, and therefore a high level of reliability is needed. In such cases it MUST be possible to provide acknowledgment by the target to the messaging system or to repeat notification until such an acknowledgement is provided if supported by the notification channel. Alternatively it must be possible for the messaging system to request such repeats.

15. Security Considerations

Notifications must be secured (when useful information is sent) and integrity should be checkable.

It should be possible to authenticate sender and prevent Denial of Service attack via notifications.

16. Normative and Informative References

[[editor's note: split into two sections, update references]]

[ABNF] D. Crocker, et al. "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997. http://www.ietf.org/rfc/rfc2234

[ANNOTATEMORE] Daboo, C., "IMAP ANNOTATEMORE Extension", work in progress, draft-daboo-imap-annotatemore-xx, (work in progress).

[GSM03.40] GSM 03.40 v7.4.0 Digital cellular telecommunications system (Phase 2+); Technical realization of the Short Message Service (SMS). ETSI 2000

[IMAP-DISC] Melnikov, A. "Synchronization operations for disconnected IMAP4 clients", draft-melnikov-imap-disc-06.txt, October 2004.

[IMAP-EVENTS] Melnikov, A. " Lemonade Inband Notifications", draft-melnikov-lemonade-imap-events-00.txt, June 17, 2006

[IMAPSIEVE] Leiba, B., "Support for Sieve in Internet Message Access

Protocol (IMAP4)", draft-ietf-lemonade-imap-sieve-0x (work in progress).

[LEMONADEPROFILE] Maes, S.H. and Melnikov A., "Lemonade Profile", draft-ietf-lemonade-profile-XX.txt, (work in progress).

[LEMONADEPROFILEBIS] Maes, S.H., Melnikov A. and D. Cridland, " LEMONADE profile bis", draft-ietf-lemonade-profile-bis-xx.txt, (work in progress).

[MANAGESIEVE] Martin, T. and A. Melnikov, "A Protocol for Remotely Managing Sieve Scripts", draft-martin-managesieve-xx.txt, (work in progress).

[MSGEVENTS] Newman, C., "Internet Message Store Events", draft-newman-lemonade-msgevent-xx.txt, (work in progress).

[NOTIFICATIONPROTOCOL] Maes, S.H., " Lemonade Notification protocol ", draft-ietf-lemonade-notification-protocol-xx.txt, (work in progress).

[OMA-EN] Open Mobile Alliance Email Notification Version 1.0, August 2002. http://www.openmobilealliance.org/tech/docs/EmailNot/OMA-Push-EMN-V1_0-20020830-C.pdf

[OMA-ME-AD] Open Mobile Alliance Mobile Email Architecture Document, (Work in progress). http://www.openmobilealliance.org/

[OMA-ME-RD] Open Mobile Alliance Mobile Email Requirement Document, (Work in progress). http://www.openmobilealliance.org/

[P-IMAP] Maes, S.H., Lima R., Kuang, C., Cromwell, R., Ha, V. and Chiu, E., Day, J., Ahad R., Jeong W-H., Rosell G., Sini, J., Sohn S-M., Xiaohui F. and Lijun Z., "Push Extensions to the IMAP Protocol (P-IMAP)", draft-maes-lemonade-p-imap-xx.txt, (work in progress).

[RECONNECT] A. Melnikov, C. Wilson, "IMAP4 Extensions for Quick Reconnect", draft-ietf-lemonade-reconnect-06.txt, May 2006

[RFC2177] Leiba, B. "IMAP4 IDLE Command", RFC 2177, June 1997. http://www.ietf.org/rfc/rfc2177.

[RFC3501] Crispin, M. "IMAP4, Internet Message Access Protocol Version 4 rev1", RFC 3501, March 2003. http://www.ietf.org/rfc/rfc3501

[SIEVE] SIEVE WG, http://www.ietf.org/html.charters/sieve-charter.html

[SIEVENOTIFICATIONS] Melnikov, A., "Sieve -- An extension for providing instant notifications", draft-ietf-sieve-notify-XX.txt, (work in progress)

[URLAUTH] Crispin, M. and Newman, C., "Internet Message Access Protocol (IMAP) - URLAUTH Extension", draft-ietf-lemonade-urlauth-XX.txt, (work in progress).

[VFOLDER] Maes, S. and et Al., "Persistent Search Extensions and Virtual Folder to the IMAP Protocol", draft-maes-lemonade-vfolder-0x, (work in progress).

[WAPWDP] Wireless Datagram Protocol, Version 14-Jun-2001, Wireless Application Protocol WAP-259-WDP- 20010614-aWAP (WDP)

17. Future Work

[1] Complete the specification tasks and editor s identified in this document:

* ^Detailed NF specifications (Sieve or no sieve)
* ^NF filter management protocol
* ^Create new MSGEVENTS draft to define mandatory event support, including new OMA required events like client LOCK_DOWN, or request the client to re-provision (including encryption keys)

[2] Map MSGEVENTS to mandatory in-band responses

[3] Determine whether CHECKPOINT style inband event queuing is needed when client is disconnected, or whether [RECONNECT] suffices (e.g. we may need a lemonade idle event draft)

[4] Possibly update MSGEVENTS, keeping what is necessary, and adding new ones (e.g. we may need a lemonade event draft starting from

[MSGEVENTS]).

[5] Review and sanitize introduction of server to server notification and possibly better integrate with structure of the text.

[6] Better relate and divide text between this draft and [NOTIFICATIONPROTOCOL]

[7] Reflect decisions on discussions of support of notification of multiple devices per user.

18. Acknowledgments

The authors want to thank all who have contributed key insight in notifications and filtering and have authored specifications or drafts used in this document, including the original work on P-IMAP [P-IMAP].

The authors want to thank the authors of the original work on Server To Server Notification Protocol Requirements (draft-ietf-lemonade-notify-s2s-00) whose material has been incorporated in the present document, in particular, Gev Decktor.

19. 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

Ray Cromwell
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Appendix A: Out-of-band SMS channel binding (INFORMATIVE appendix)

One method for delivering wake-up notifications is by pushing the notification payload as a binary SMS message. Upon receiving an SMS, a client would then parse the payload, determine if it is a email notification or some other SMS message, and process the message appropriately.

This has the unfortunate side effect of forcing the client to parse every message trying to sense what kind of message it is. The proposed mechanism to fix this is to utilize the binary

SMS User Data Header (UDH) to specify a destination port, according to the Application Port

Addressing Scheme in [GSM03.40] or alternatively, on CDMA networks, to use the WAP WDP mapping to GSM SMS [WAPWDP].

Although any port number is usable, it might make sense to use port 143 for consistency, which is the IANA IMAP port. Thus, OMA EMN or extended format notifications should be sent to port 143 via GSM SMS or WAP WDP. The client upon receiving the SMS will check the port number, and if the port is the right port, the message will be routed to the appropriate client application for processing.

Because such mechanisms are network specific, a server should determine if a port specific SMS or WAP WDP mapping can be used based on knowledge of the device / network or on strategies that determine if the device reacts to such notifications. However, a client may also declare it / selecting the out-of-band notification channel as GSMSMS or WAPWDP as for any other notification channel.

Appendix B: Changes from Previous Versions

THIS SECTION TO BE REMOVED PRIOR TO PUBLICATION.

version -04:
* Update dates, slight reformatting, add editor's note for References

version -03:
* Updated examples to use new METADATA syntax
* Drop CLEARIDLE and reference A. Melnikov's [IMAP-EVENTS]

* XEMN notification format extended to with event and view attributes
* View filter is a work in progress. Several proposals are being discussed, so the draft has been revised to try and capture high level requirements (e.g. out of band notifications must be able to identify which view an event occurred for)
* Added notification protocol details and reference

version -02:
* LPROVISION/LGETPREFS/LSETPREFS removed in favor of mailbox annotations
* Updated inband notification section to include discussion of [CLEARIDLE] and [MSGEVENTS]
* EMN payload clarified for both wakeup and extended formats.
* Some reference clean-up
* Add server to server notifications based on the expired draft draft-ietf-lemonade-notify-s2s-00.

version -01:
* Move SMS / WAP examples to an informative appendix.
* Restrict the exchange of keys via LPROVISION to secure exchanges.
* Differentiate ANNOTATE from LPROVISION on that basis.

versin -00:
* Initial release

Intellectual Property Statement

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

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository

at http://www.ietf.org/ipr.

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

Full Copyright Statement

Copyright (C) The IETF Trust (2007).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property Statement

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

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

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

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.