Lemonade                                                     S. H. Maes
Internet Draft: IMAP HTTP Binding                           R. Cromwell
N. Mitra
(Editors)

Document: draft-maes-lemonade-http-binding-04
Expires: July 2006                                         January 2006

IMAP and SMTP HTTP Binding

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

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

As part of the LEMONADE work to define extensions to the IMAPv4 Rev1
protocol [RFC3501] and SMTP that provide optimizations in a variety
of settings, this document describes an alternative, optional
binding for IMAPv4 and SMTP showing how HTTP can be used to transfer
commands and responses. This binding is intended to facilitate the
use of IMAP and SMTP in deployments involving a variety of
intermediaries. A binding to SOAP is also provided.
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], [RFC821], and related documents.

Table of Contents

Status of this Memo............................................... 1
Copyright Notice.................................................. 1
Abstract.......................................................... 1
Conventions used in this document............................... 2
Table of Contents.................................................. 2
1. Introduction and motivation................................... 2
2. Techniques for binding over HTTP............................. 4
   2.1. Tunneling Approaches.................................... 4
      2.1.1. Non-Persistent HTTP for In-response Connectivity Mode........................................ 6
      2.1.2. Using Persistent HTTP/HTTPS + Chunked Transfer Encoding for In-band Connectivity Mode..... 7
      2.1.3. Using HTTP Connect.................................. 8
      2.1.4. Using HTTP as a binding for SMTP................... 9
   2.2. Using SOAP (Web Services) as a binding for IMAP........ 9
3. Security Considerations....................................... 11
4. References..................................................... 11
5. Future Work.................................................... 13
6. Version History................................................ 13
Acknowledgments.................................................. 14
Authors Addresses................................................ 14
Intellectual Property Statement.................................. 16
Disclaimer of Validity........................................... 16
Copyright Statement............................................. 16

1. Introduction and motivation

Maes                     Expires Â July 2006
As part of the LEMONADE goal to define extensions to the IMAPv4 Rev1 protocol [RFC3501] for providing optimizations in a variety of settings, this document describes how HTTP can optionally be used to transfer IMAP and SMTP commands and responses. This binding is intended to facilitate the use of IMAP and SMTP in deployments involving a variety of intermediaries, and offers a standardized alternative to de facto proprietary implementations of such a feature.

The need for an optional HTTP binding is driven by the needs of the mobile network operator community (see [MEMAIL][OMA-ME-RD]), where the reuse of an existing and well-understood technology will allow operators to apply their experience in solving practical deployment issues. Specifically, HTTP allow operators to reuse a similar setup and model that is already used for many other similar and related services, such as certain proprietary push e-mail and synchronization offerings, OMA Data Synchronization, Web services and Web access.

Using HTTP/HTTPS can simplify deployment in a corporate network through the potential use of a reverse proxy to achieve end-to-end encryption. This also has the advantage of not requiring changes to any firewall configurations and reduces the concerns that this often presents to corporation. In general the solution is compatible with any existing firewall. A reverse proxy can also support deployment models that offer roles to other service providers in the value chains, as discussed in [OMA-ME-AD].

The confidentiality, integrity, and compression capabilities used with HTTP and already implemented in a wide range of existing mobile device, which be also be reused.

Studies have also shown that a persistent HTTP session has usually proven more resilient than an IMAP IDLE over TCP connection over an unreliable bearer such as a GPRS-based mobile network.

The use of HTTP as an underlying protocol for other application protocols has received much attention (see [RFC3205]). In particular, the concern exists that this circumvents firewall security policies. Another concern is the potential misuse or neglect of HTTP semantics by the application protocol that uses HTTP as a substrate.

Note that if the suppression of IMAP (or indeed any other application) traffic on HTTP/HTTPS is an issue, firewall administrators can still prevent such passage and this can provide incentives to re-configure firewalls to allow solutions on other transports (e.g. TLS) or offer the HTTP-based solution using another provisioned port (e.g. manually, out of band or via instructions like XGETLPREFS (see [NOTIFICATIONS])). The aim, therefore, is to allow for the use of this solution in the widest possible set of...
circumstances by codifying a standard way to do so that works with existing, deployed (i.e., HTTP only) firewalls, while explicitly allowing the possibility of detecting and filtering such traffic in deployments using the HTTP Content-Type in deployments where this is not permitted.

A SOAP binding is also described.

2. Techniques for binding over HTTP

There are two approaches described below for binding IMAP over HTTP. The first approach shows how to tunnel regular IMAP requests and responses over HTTP using POST. The second method proposes a syntactic change which recodes IMAP requests and responses as SOAP documents, and IMAP commands as SOAP methods.

<EditorÂs note: More approaches and a rationalization of the possible approach will be added later.>

2.1. Tunneling Approaches

To use HTTP/HTTPS as the transfer protocol for IMAP commands and responses between the IMAP client and server, the client MUST send an HTTP POST request to the server, and embed IMAP commands (commands to an IMAPv4 Rev1 server or IMAP servers supporting Lemonade extensions) in the body of the request. A server MUST reject a HTTP GET request from the client. The content-type header of the POST request MUST be set to "application/vnd.lemonade". Multiple IMAP commands may be included in one POST request. In general, the HTTP server is expected to preserve session state between HTTP commands to the best of its ability, therefore the client does not need to reauthenticate and reissue a SELECT until it receives an (IMAP) error response showing that it is not authenticated.

In what follows, the term Lemonade client/server is used to refer to a client/server that supports both IMAPv4 Rev1 as well as any LEMONADE extensions.

When the HTTP binding is used, the Lemonade server listens on whatever port has been configured for this.

The following is an example of a possible Lemonade HTTP request:

```plaintext
POST /lemonadePath HTTP/1.1 <CRLF>
Content-Type: application/vnd.lemonade <CRLF>
[other headers]
<CRLF>
```
The Lemonade command MUST be plain text (7bit).

Multiple Lemonade commands MAY be sent on the same request. Thus Lemonade commands must be tagged. The client must be able to deal with recovering from errors when commands are batched. See RFC2442 Batch SMTP for a further discussion.

The Content-Type header is the only HTTP headers that MUST be sent to a Lemonade server. Other headers such as Cache-Control MAY be included.

When the Lemonade server sends back a response it MUST be in the following format:

```
HTTP/1.1 <HTTP Status Code> <CRLF>
Content-Type: text/plain <CRLF>
<CRLF>
[<untagged responses>]
<tag> SP <Lemonade Server response> <CRLF>
[<untagged responses>]
<tag> SP <Lemonade Server response> <CRLF>
```

Notes:
The Lemonade Server uses the following HTTP status codes, and what each code indicates is given below:
- **200**
  - This indicates normal execution of the Lemonade commands from a IMAP perspective. The client should further parse the response body to get the tagged responses to the commands and process those accordingly.
- **500**
  - This indicates that at least one command caused an internal server error, meaning the Lemonade Server failed to execute the command. In conforming to HTTP semantics, this means the IMAP server responses such as BAD or NO IMAP generate a HTTP 500 response code.

When using HTTP to transfer IMAP commands and responses, the client SHOULD utilize built-in features of HTTP to their advantage. For example, the client SHOULD use HTTPS instead of HTTP whenever possible, since HTTPS has built in encryption and MAY have compression capabilities. STARTTLS should not be needed in this case, as it just requires additional overhead without any additional benefit.
HTTP can be used in both in-response and in-band modes. Details about these transport modes are given in the following two subsections.

2.1.1. Non-Persistent HTTP for In-response Connectivity Mode

If the client uses a traditional HTTP connection (either by establishing a different socket for each HTTP request to the Lemonade server, or by reusing the same socket for all HTTP requests, but sending each request under its own header), it has in-response connectivity to the server. The client can issue as many commands as it would like in one HTTP request to the server, and the server responds by sending back one HTTP response with all the responses to all the commands in the HTTP request. With this connectivity mode, the IDLE command cannot be issued. Other commands that use a continuation response cannot be issued unless the LITERAL+ [RFC2088] extension is supported.

In order for the server to identify separate HTTP requests as belonging to the same session, an in-response HTTP client needs to accept cookies. A session-id is passed in the cookie to identify the session.

Thus, the headers for a HTTP In-response Response after the client has issued its first HTTP request to the server.

```
HTTP/1.1 <HTTP Status Code> <CRLF>
Content-Type: text/plain <CRLF>
Set-Cookie:JSESSIONID=94571a8530d91e1913bfydafa;
p=path=/lemonade<CRLF>
<CRLF>
[<untagged responses>]
<tag> SP <Lemnade Server response> <CRLF>
[<untagged responses>]
<tag> SP <Lemonade Server response> <CRLF>
```

The client must then save this cookie and send it back to the server with the next request in order for the server to reattach these commands to the same session as the previous commands.

```
POST /lemonadePath HTTP/1.1 <CRLF>
Content-Type: application/vnd.lemonade <CRLF>
Cookie: JSESSIONID=94571a8530d91e1913bfydafa
[other headers]
<CRLF>
<tag> SP <Lemonade command> <CRLF>
[<tag> SP <Lemonade command> <CRLF>]
```
2.1.2. Using Persistent HTTP/HTTPS + Chunked Transfer Encoding for In-band Connectivity Mode

It is possible to use persistent HTTP or persistent HTTPS plus chunked-transfer-encoding so that the server can instantly send notifications to the client while a session is open. The client needs to open a persistent connection and keep it active. In this case, the HTTP headers must be sent the first time the client device opens the connection to the Lemonade Server and these headers MUST set the transfer coding to be chunk-encoded [RFC2616, Sec. 3.6.1]. All subsequent client-server requests are written to the open connection, without needing any additional headers negotiations. The server can use this open channel to push events to the client device at any time. In this case, the client SHOULD NOT accept cookies.

The client must send the HTTP headers one time only:

    POST /lemonadeServletPath HTTP/1.1 <CRLF>
    Content-Type: application/vnd.lemonade <CRLF>
    Connection: keep-alive <CRLF>
    Pragma: no-cache <CRLF>
    Transfer-Encoding: chunked <CRLF>

The server responds with the following header:

    HTTP/1.1 <HTTP Status Code> <CRLF>
    Cache-Control: private
    Keep-Alive: timeout=15, max=100 (or other suitable setting)
    Connection: Keep-Alive
    Transfer-Encoding: chunked
    Content-Type: text/plain

Then the client can send a command anytime it wants with the following format:

    <length of Lemonade command, including bytes in CRLF> <CRLF>
    <tag> SP <Lemonade command> <CRLF>
    <CRLF>

And example of an actual client command is:

    e <CRLF>
    2 CAPABILITY<CRLF>
    <CRLF>

The server responds to each command with as many untagged responses as needed, and one tagged response, where each response is in the format that follows:
An actual Server response might be:

d5 <CRLF>
* CAPABILITY IMAP4REV1 AUTH=LOGIN NAMESPACE SORT MULTIAPPEND
LITERAL+ UIDPLUS IDLE XORACLE X-ORACLE-LIST X-ORACLE-COMMENT X-
ORACLE-QUOTA X-ORACLE-PREF X-ORACLE-MOVE X-ORACLE-DELETE ACL X-
ORACLE-PASSWORD LDELIVER LZIP LCONVERT LFILTER LSETPREF LGETPREF
<CR><LF>   <CR><LF>
1b <CR><LF>
 2 OK CAPABILITY completed <CR><LF>
<CR><LF>

Note however that the HTTP protocol is in general not meant to be
used in such a way. To maintain such an open channel might be a
practical challenge to proxies/firewalls, which might not forward the
requests chunk by chunk to the server, and meanwhile route responses
back to the client chunk by chunk. Consequently the session closes.
Chunked transfer encoding requests MAY not be honored by an HTTP
server. In cases where such requests are denied, the client should be
prepared to use the non-chunked encoding technique from section 2.1

The same challenges exist for TCP session.

In any case, the session can be automatically started again by the
client after a lost connection or by the server through out-of-band;
after some defined time-out.

2.1.3. Using HTTP Connect

If a HTTP proxy server is available to the client which supports the
HTTP CONNECT method, and the IMAP server the user wishes to reach
allows external connections outside the destination networkÀs
firewall, the client may wish to tunnel a regular TCP connection
through the HTTP proxy.

See [LUOTONEN] or section 5.2 of [RFC2817] for a detailed
description of the technique. Note that HTTP Proxy servers may not
honor all CONNECT requests, and may in fact, limit CONNECT requests
to a small number of common ports, such as 80, 443, 8080, etc. It is
advised that networks wishing to allow their users to use this
feature allow clients within their network to CONNECT to ports 25,
143, 587, and 993.
2.1.4. Using HTTP as a binding for SMTP

All of the techniques described in sections 2.1, 2.2, and 2.3 may be used for SMTP as well. The only difference between IMAP and SMTP will be the HTTP URL used. Servers implementing the HTTP binding are expected to differentiate between IMAP and SMTP protocol bodies via the URL.

2.2. Using SOAP (Web Services) as a binding for IMAP

The SOAP binding attempts to map IMAP commands to SOAP methods, and IMAP data types and grammar (atoms, lists, etc.) to document-literals supplied as the soap body. This is essentially a tunneling technique with a syntactic change. The following general encoding rules are proposed:

IMAP commands are translated into SOAP methods of the same name, e.g. the FETCH command becomes the FETCH SOAP method name. (UID FETCH is mapped to UID_FETCH).

SOAP document literal style is used

Terms in the IMAP grammar which represent atoms become elements. (e.g. FLAGS becomes <FLAGS/>) Flags are stripped of leading backslash and uppercased.

Non-terminals which of an ATOM followed by a single parameter are represented as a non-empty element containing that parameter. (e.g. CHARSET foo becomes <CHARSET>foo</CHARSET>, or SENTBEFORE date becomes <SENTBEFORE>date</SENTBEFORE>

Lists are represented as <L> </L> containing zero or more elements (including other <L>s)

Unless otherwise defined, if a particular keyword is followed by more than one value, each value is encoded as <P>value</P> as placed as a child element. E.g. APPEND mailbox SP flaglist SP literal becomes <APPEND><P>mailbox</P><P><L><ANSWERED/><DRAFT/></L></P></APPEND>

Continuation responses and requests are encapsulated as <C>data</C>

Literals are encapsulated as <T>text</T> or <B>binary</B>

Unsolicited responses are encapsulates as <U>response</U>

The partial specifier is <P>offset.length</P>

The section specifier is <SECTION>Â</SECTION>

A sequence set is wrapped as <SEQUENCE>sequence-set</SEQUENCE>

The IMAP response is encoded in <RESP>response</RESP>

Any responses which start with a number followed by an ATOM are encoded as <ATOM>number</ATOM>

The following is an example encoding:

C: a1 FETCH 1:5,9 BODY[1.1.CONVERT(ÂTEXT/PLAINÂ)]<1024.2048>

Becomes
<FETCH>
  <SEQUENCE>1:5, 9</SEQUENCE>
  <BODY>
    <SECTION>
      <P>1.1.CONVERT(ÂTEXT/PLAINÂ)</P>
    </SECTION>
    <P>1024.2048</P>
  </BODY>
</FETCH>

Which would then be invoked on a Web Service via the SOAPMethodName ÂFETCHÂ. The expected response would be zero or more <U> elements containing <FETCH> elements which encode the returned data.

These rules are by no means complete and exhaustive, and more stringent encoding rules are needed to encompass the full range of IMAP extended ABNF. The above rules are provided as a starting point.

SOAP by itself adds considerable overhead to requests, so it would not be recommended without some form of compression or compact encoding such as ÂFast Web ServicesÂ (X.695 ÂASN.1 Support for SOAP, Web Services and the XML Information SetÂ)[X.695]. However, SOAP may provide some benefits over raw HTTP for those who have existing investments in SOAP infrastructure.

As a final note, the above usage once again, assumes that the soap server is not stateless and uses HTTP cookies to preserve IMAP session state between requests.

HereÂs an example session side by side with IMAP syntax(SOAP envelop not shown):

C-SOAP: <LOGIN><P>username</P><P>password</P>
C-IMAP: a1 LOGIN username password

S-SOAP: <RESP><OK>LOGIN Ok</OK>
S-IMAP: * OK LOGIN Ok

C-SOAP: <SELECT>INBOX</SELECT>
C-IMAP: a2 SELECT INBOX

S-SOAP: <RESP>
  <U><FLAGS><L><ANSWERED/><DRAFT/><FLAGGED/><SEEN/></L></FLAGS></U>
  <U><OK><PERMANENTFLAGS><L><ANSWERED/><DRAFT/><FLAGGED/><SEEN/></L>
    <PERMANENTFLAGS></OK></U>
  <U><EXISTSS>1234</U>
  <U><RECENT>0</U>
  <U><OK><UIDVALIDITY>12345678</UIDVALIDITY></OK></U>
</RESP>
3. Security Considerations

HTTP binding has the same security requirements as IMAP when using an in-response or inband connectivity mode.

The HTTPS protocol can be used to provide end-to-end security.

Proxy-based implementations may still require payload encryption for end-to-end security.

Caching is a concern. The client SHOULD use the HTTP Cache-Control directive (no-cache, no-store, must-revalidate, or combinations thereof) to inform proxy servers, origin servers, and client libraries not to cache or store the HTTP response. To deal with HTTP 1.0 servers that may exist in the network, Pragma: no-cache should be used as well.

Attacks on HTTP sessions and the HTTP server may also be a concern, since the HTTP server is maintaining an authenticated session to the IMAP server on behalf of the user in most cases.

Firewall administrators wishing to block stealth deployments of HTTP IMAP bindings may block HTTP requests with Content-Type application/vnd.lemonade via an application level firewall.

4. References


[LUOTONEN] Luotonen, A., Tunneling TCP based protocols through Web proxy servers, draft-luotonen-web-proxy-tunneling-01.txt, August 1998


[X.695] X.695 ÂASN.1 Support for SOAP, Web Services and the XML Information SetÂ, ITU/ISO
http://java.sun.com/developer/technicalArticles/WebServices/fastWS/

5. Future Work

[1] Should an OPTIONS HTTP request be supported to allow a client to probe HTTP binding capabilities, such as which protocol a given URL is bound to, or whether chunking is supported?

[2] Should separate content types exist for IMAP and SMTP since the entity body in the HTTP request is different?

[3] Standardizing the form of the URL for the binding may permit firewall administrations to impose better filtering.

[4] Investigate WebDAV binding and any DAV extensions (if any) needed

[5] Investigate REST binding

[6] Present a detailed formalism for the possible methods:

- HTTP CONNECT
- HTTP POST (disconnected)
- HTTP POST + Chunked (persistent)
- SOAP
- DAV

6. Version History

Release 04
Added SMTP and Future Work.
Clarified caching policy.
Initial SOAP binding

Release 03
Removed material on Notifications and connectivity models
Updated introduction with motivation
Editorial corrections

Release 02
New section that allows to select the HTTP URL.
New section 4 to motivate the introduction of an HTTP binding.
Editorial updates

Release 01
Detail updates of the text throughout the document following lessons learned so far in P-IMAP 07 [P-IMAP].
Release 00

Acknowledgments

The authors want to thank all who have contributed key insight and extensively reviewed and discussed the concepts of HTTP Bindings and its early introduction in P-IMAP [P-IMAP].

Authors Addresses

Stephane H. Maes
Oracle Corporation
500 Oracle Parkway
M/S 40p634
Redwood Shores, CA 94065
USA
Phone: +1-650-607-6296
Email: stephane.maes@oracle.com

Rafiul Ahad
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Eugene Chiu
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

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

Jia-der Day
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Wook-Hyun Jeong
Samsung Electronics, CO., LTD
416, Maetan-3dong, Yeongtong-gu,
Suwon-city, Gyeonggi-do,
Korea 442-600
Tel: +82-31-279-8289
E-mail: wh75.jeong@samsung.com

Chang Kuang
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Rodrigo Lima
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065
USA

Nilo Mitra
Ericsson
Tel: +1 212-843-8451
Email: nilo.mitra@ericsson.com

Gustaf Rosell
Sony Ericsson
P.O. Box 64
SE-164 94 Kista,
Sweden
Tel: +46 8 508 780 00

Jean Sini
6480 Via Del Oro
San Jose, CA 95119
USA

Sung-Mu Son
LG Electronics
Mobile Communication Technology Research Lab.
Tel: +82-31-450-1910
E-Mail: sungmus@lge.com

Fan Xiaohui
Product Development Division
R&D CENTER
CHINA MOBILE COMMUNICATIONS CORPORATION (CMCC)
ADD: 53A, Xibianmennei Ave.,Xuanwu District,
Beijing,100053
China
TEL:+86 10 66006688 EXT 3137

Zhao Lijun
CMCC R&D
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.