Internet Message Access Protocol

In computing, the **Internet Message Access Protocol** (**IMAP**) is an Internet standard protocol used by email clients to retrieve email messages from a mail server over a TCP/IP connection.[1] IMAP is defined by RFC 3501.

IMAP was designed with the goal of permitting complete management of an email box by multiple email clients, therefore clients generally leave messages on the server until the user explicitly deletes them. An IMAP server typically listens on port number 143. IMAP over SSL (**IMAPS**) is assigned the port number 993.

Virtually all modern e-mail clients and servers support IMAP, which along with the earlier POP3 (Post Office Protocol) are the two most prevalent standard protocols for email retrieval.[2] Many webmail service providers such as Gmail, Outlook.com and Yahoo! Mail also provide support for both IMAP and POP3.

Contents

E-mail protocols

The Internet Message Access Protocol is an Application Layer Internet protocol that allows an e-mail client to access e-mail on a remote mail server. The current version is defined by RFC 3501. An IMAP server typically listens on well-known port 143, while IMAP over SSL (IMAPS) uses 993.

Incoming e-mail messages are sent to an e-mail server that stores messages in the recipient's e-mail box. The user retrieves the messages with an e-mail client that uses one of a number of e-mail retrieval protocols. While some clients and servers preferentially use vendor-specific, proprietary protocols,[3] almost all support POP and IMAP for retrieving e-mail - allowing many free choice between many e-mail clients such as Pegasus Mail or Mozilla Thunderbird to access these servers, and allows the clients to be used with other servers.

E-mail clients using IMAP generally leave messages on the server until the user explicitly deletes them. This and other characteristics of IMAP operation allow multiple clients to manage the same mailbox. Most e-mail *clients* support IMAP in addition to Post Office Protocol (POP) to retrieve messages.[4] IMAP offers access to the mail storage. Clients may store local copies of the messages, but these are considered to be a temporary cache.

History

IMAP was designed by Mark Crispin in 1986 as a remote access mailbox protocol, in contrast to the widely used POP, a protocol for simply retrieving the contents of a mailbox.

It went through a number of iterations before the current VERSION 4rev1 (IMAP4), as detailed below:

Original IMAP

The original *Interim Mail Access Protocol* was implemented as a Xerox Lisp machine client and a TOPS-20 server.

No copies of the original interim protocol specification or its software exist.[5][6] Although some of its commands and responses were similar to IMAP2, the interim protocol lacked command/response tagging and thus its syntax was incompatible with all other versions of IMAP.

IMAP2

The interim protocol was quickly replaced by the *Interactive Mail Access Protocol* (IMAP2), defined in RFC 1064 (in 1988) and later updated by RFC 1176 (in 1990). IMAP2 introduced the command/response tagging and was the first publicly distributed version.

IMAP3

IMAP3 is an extremely rare variant of IMAP.[7] It was published as RFC 1203 in 1991. It was written specifically as a counter proposal to RFC 1176, which itself proposed modifications to IMAP2.[8] IMAP3 was never accepted by the marketplace.[9][10] The IESG reclassified RFC1203 "Interactive Mail Access Protocol - Version 3" as a Historic protocol in 1993. The IMAP Working Group used RFC1176 (IMAP2) rather than RFC1203 (IMAP3) as its starting point.[11][12]

IMAP2bis

With the advent of MIME, IMAP2 was extended to support MIME body structures and add mailbox management functionality (create, delete, rename, message upload) that was absent from IMAP2. This experimental revision was called IMAP2bis; its specification was never published in non-draft form. An internet draft of IMAP2bis was published by the IETF IMAP Working Group in October 1993. This draft was based upon the following earlier specifications: unpublished *IMAP2bis.TXT* document, RFC1176, and RFC1064 (IMAP2).[13] The *IMAP2bis.TXT* draft documented the state of extensions to IMAP2 as of December 1992.[14] Early versions of Pine were widely distributed with IMAP2bis support[7] (Pine 4.00 and later supports IMAP4rev1).

IMAP4

An IMAP Working Group formed in the IETF in the early 1990s took over responsibility for the IMAP2bis design. The IMAP WG decided to rename IMAP2bis to IMAP4 to avoid confusion.

Advantages over POP

Connected and disconnected modes

When using POP, clients typically connect to the e-mail server briefly, only as long as it takes to download new messages. When using IMAP4, clients often stay connected as long as the user interface is active and download message content on demand. For users with many or large messages, this IMAP4 usage pattern can result in faster response times.

Multiple simultaneous clients

The POP protocol requires the currently connected client to be the only client connected to the mailbox. In contrast, the IMAP protocol specifically allows simultaneous access by multiple clients and provides mechanisms for clients to detect changes made to the mailbox by other, concurrently connected, clients. See for example RFC3501 section 5.2 which specifically cites "simultaneous access to the same mailbox by multiple agents" as an example.

Access to MIME message parts and partial fetch

Usually all Internet e-mail is transmitted in MIME format, allowing messages to have a tree structure where the leaf nodes are any of a variety of single part content types and the non-leaf nodes are any of a variety of multipart types. The IMAP4 protocol allows clients to retrieve any of the individual MIME parts separately and also to retrieve portions of either individual parts or the entire message. These mechanisms allow clients to retrieve the text portion of a message without retrieving attached files or to stream content as it is being fetched.

Message state information

Through the use of flags defined in the IMAP4 protocol, clients can keep track of message state: for example, whether or not the message has been read, replied to, or deleted. These flags are stored on the server, so different clients accessing the same mailbox at different times can detect state changes made by other clients. POP provides no mechanism for clients to store such state information on the server so if a single user accesses a mailbox with two different POP clients (at different times), state information—such as

whether a message has been accessed—cannot be synchronized between the clients. The IMAP4 protocol supports both predefined system flags and client-defined keywords. System flags indicate state information such as whether a message has been read. Keywords, which are not supported by all IMAP servers, allow messages to be given one or more tags whose meaning is up to the client. IMAP keywords should not be confused with proprietary labels of web-based e-mail services which are sometimes translated into IMAP folders by the corresponding proprietary servers.

Multiple mailboxes on the server

IMAP4 clients can create, rename, and/or delete mailboxes (usually presented to the user as folders) on the server, and copy messages between mailboxes. Multiple mailbox support also allows servers to provide access to shared and public folders. The *IMAP4 Access Control List (ACL) Extension* (RFC 4314) may be used to regulate access rights.

Server-side searches

IMAP4 provides a mechanism for a client to ask the server to search for messages meeting a variety of criteria. This mechanism avoids requiring clients to download every message in the mailbox in order to perform these searches.

Built-in extension mechanism

Reflecting the experience of earlier Internet protocols, IMAP4 defines an explicit mechanism by which it may be extended. Many IMAP4 extensions to the base protocol have been proposed and are in common use. IMAP2bis did not have an extension mechanism, and POP now has one defined by RFC 2449 (https://tools.ietf.org/html/rfc2449).

Disadvantages

While IMAP remedies many of the shortcomings of POP, this inherently introduces additional complexity. Much of this complexity (e.g., multiple clients accessing the same mailbox at the same time) is compensated for by server-side workarounds such as Maildir or database backends.

The IMAP specification has been criticised for being insufficiently strict and allowing behaviours that effectively negate its usefulness. For instance, the specification states that each message stored on the server has a "unique id" to allow the clients to identify messages they have already seen between sessions. However, the specification also allows these UIDs to be invalidated with no restrictions, practically defeating their purpose.[15]

Unless the mail storage and searching algorithms on the server are carefully implemented, a client can potentially consume large amounts of server resources when searching massive mailboxes.

IMAP4 clients need to maintain a TCP/IP connection to the IMAP server in order to be notified of the arrival of new mail. Notification of mail arrival is done through in-band signaling, which contributes to the complexity of client-side IMAP protocol handling somewhat.[16] A private proposal, push IMAP, would extend IMAP to implement push e-mail by sending the entire message instead of just a notification. However, push IMAP has not been generally accepted and current IETF work has addressed the problem in other ways (see the Lemonade Profile for more information).

Unlike some proprietary protocols which combine sending and retrieval operations, sending a message and saving a copy in a server-side folder with a base-level IMAP client requires transmitting the message content twice, once to SMTP for delivery and a second time to IMAP to store in a sent mail folder. This is addressed by a set of extensions defined by the IETF Lemonade Profile for mobile devices: URLAUTH (RFC 4467 (https://tools.ietf.org/html/rfc4467)) and CATENATE (RFC 4469 (https://tools.ietf.org/html/rfc4469)) in IMAP and BURL (RFC 4468 (https://tools.ietf.org/html/rfc4468)) in SMTP-SUBMISSION. In addition to this, Courier Mail Server offers a non-standard method of sending using IMAP by copying an outgoing message to a dedicated outbox folder.[17]

Security

To cryptographically protect IMAP connections, IMAPS on TCP port 993 can be used, which utilizes TLS. As of RFC 8314, this is the recommended mechanism.

Alternatively, STARTTLS can be used to provide secure communications between the MUA communicating with the MSA or MTA implementing the SMTP Protocol.

Dialog example

This is an example IMAP connection as taken from RFC 3501 section 8 (https://tools.ietf.org/html/rfc3501#section-8):

```
C: <open connection>
S:   * OK IMAP4rev1 Service Ready
C:   a001 login mrc secret
S:   a001 OK LOGIN completed
C:   a002 select inbox
S:   * 18 EXISTS
S:   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S:   * 2 RECENT
S:   * OK [UNSEEN 17] Message 17 is the first unseen message
S:   * OK [UIDVALIDITY 3857529045] UIDs valid
S:   a002 OK [READ-WRITE] SELECT completed
C:   a003 fetch 12 full
S:   * 12 FETCH (FLAGS (\Seen) INTERNALDATE "17-Jul-1996 02:44:25 -0700"
      RFC822.SIZE 4286 ENVELOPE ("Wed, 17 Jul 1996 02:23:25 -0700 (PDT)"
      "IMAP4rev1 WG mtg summary and minutes"
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      ((NIL NIL "imap" "cac.washington.edu"))
      ((NIL NIL "minutes" "CNRI.Reston.VA.US")
      ("John Klensin" NIL "KLENSIN" "MIT.EDU")) NIL NIL
      "<B27397-0100000@cac.washington.edu>")
      BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028
      92))
S:   a003 OK FETCH completed
C:   a004 fetch 12 body[header]
S:   * 12 FETCH (BODY[HEADER] {342}
S:   Date: Wed, 17 Jul 1996 02:23:25 -0700 (PDT)
S:   From: Terry Gray <gray@cac.washington.edu>
S:   Subject: IMAP4rev1 WG mtg summary and minutes
S:   To: imap@cac.washington.edu
S:   cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@MIT.EDU>
S:   Message-Id: <B27397-0100000@cac.washington.edu>
S:   MIME-Version: 1.0
S:   Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:   )
S:   a004 OK FETCH completed
C    a005 store 12 +flags \deleted
S:   * 12 FETCH (FLAGS (\Seen \Deleted))
S:   a005 OK +FLAGS completed
C:   a006 logout
S:   * BYE IMAP4rev1 server terminating connection
S:   a006 OK LOGOUT completed
```

See also

- List of mail server software
- Comparison of email clients
- Comparison of mail servers
- IMAP IDLE
- JSON Meta Application Protocol (JMAP)
- Post Office Protocol (POP)
- Push-IMAP
- Simple Mail Access Protocol
- Simple Mail Transfer Protocol
- Webmail

References

1. Dean, Tamara (2010). *Network+ Guide to Networks* (https://books.google.com/?id=UD0h_GqgbHgC&printsec=frontcover&dq=network%2B+guide+to+networks#v=onepage). Delmar. p. 519. ISBN 978-1423902454.
2. Komarinski, Mark (2000). *Red Hat Linux System Administration Handbook* (https://books.google.com/?id=UD0h_GqgbHgC&printsec=frontcover&dq=network%2B+guide+to+networks#v=onepage). Prentice Hall. p. 179. ISBN 1423902459.
3. For example, Microsoft's Outlook client uses MAPI, a Microsoft proprietary protocol, to communicate with a Microsoft Exchange Server. IBM's Notes client works in a similar fashion when communicating with a Domino server.
4. Mullet, Diana (2000). *Managing IMAP*. O'Reilly. p. 25. ISBN 0-596-00012-X.
5. Crispin, Mark (13 February 2012). "Re: [imap5] Designing a new replacement protocol for IMAP" (http://www.ietf.org/mail-archive/web/imap5/current/msg00317.html). *imap5* (Mailing list). alpine.OSX.2.00.1202131243200.38441@hsinghsing.panda.com. Retrieved 26 November 2014. "Knowledge of the original IMAP (before IMAP2) exists primarily in my mind as all the original IMAP specifications and implementations were replaced with IMAP2."
6. Service Name and Transport Protocol Port Number Registry (http://www.iana.org/assignments/service-names). Iana.org (2013-07-12). Retrieved on 2013-07-17.
7. "RFC 2061 - IMAP4 COMPATIBILITY WITH IMAP2BIS" (http://tools.ietf.org/html/rfc2061). IETF. 1996. Retrieved 2010-08-21.
8. "INTERACTIVE MAIL ACCESS PROTOCOL - VERSION 3" (http://tools.ietf.org/html/rfc1203). IETF. 1991. Retrieved 2010-08-21.
9. "IMAP2, IMAP2bis, IMAP3, IMAP4, IMAP4rev1 (LAN Mail Protocols)" (http://stason.org/TULARC/networking/lans-mail-protocols/03-IMAP2-IMAP2bis-IMAP3-IMAP4-IMAP4rev1-LAN-Mail-Protoc.html). Retrieved 2010-08-21.
10. "IMAP Overview, History, Versions and Standards" (http://www.tcpipguide.com/free/t_IMAPOverviewHistoryVersionsandStandards-3.htm). Retrieved 2010-08-21.
11. "Protocol Action: Interactive Mail Access Protocol — Version 3 to Historic (IETF mail archive)" (http://www.ietf.org/mail-archive/web/ietf/current/msg01656.html). 1993. Retrieved 2010-08-21.
12. "Innosoft and POP/IMAP protocols? (mail archive)" (http://www.pmdf.process.com/ftp/info-pmdf/aug.1993?httpd=content&type=text/plain;%20charset%3DISO-8859-1). 1993. Retrieved 2010-08-21.
13. "INTERACTIVE MAIL ACCESS PROTOCOL - VERSION 2bis (Internet Draft)" (http://tools.ietf.org/html/draft-ietf-imap-imap2bis-02). IETF. 1993. Retrieved 2010-08-21.

14. "IMAP2BIS -- EXTENSIONS TO THE IMAP2 PROTOCOL (DRAFT)" (https://web.archive.org/web/20110718192409/http://ftp.zcu.cz/pub/network/imap/old/IMAP2bis.TXT). 1992. Archived from the original (http://ftp.zcu.cz/pub/network/imap/old/IMAP2bis.TXT) on 2011-07-18. Retrieved 2010-08-21.
15. "IMAP implementation in Sup, an e-mail client written in Ruby" (https://web.archive.org/web/20071212234041/http://sup.rubyforge.org/svn/trunk/lib/sup/imap.rb). rubyforge.com. Archived from the original (http://sup.rubyforge.org/svn/trunk/lib/sup/imap.rb) on 2007-12-12. Retrieved 2011-02-22.
16. "IMAP IDLE: The best approach for 'push' e-mail" (http://www.isode.com/whitepapers/imap-idle.html). Isode.com. Retrieved 2009-07-30.
17. "Courier-IMAP: Sending mail via an IMAP connection" (http://www.courier-mta.org/imap/INSTALL.html#imapsend). Double Precision, Inc. Retrieved 2013-09-24.

Further reading

- Crispin, Mark (1988–2016). "Ten Commandments of How to Write an IMAP client" (https://web.archive.org/web/20160829131559/http://www.washington.edu/imap/documentation/commndmt.txt.html). University of Washington. Archived from the original (https://www.washington.edu/imap/documentation/commndmt.txt.html) on 2016-08-29. Retrieved 2018-11-02.
- Heinlein, P; Hartleben, P (2008). *The Book of IMAP: Building a Mail Server with Courier and Cyrus*. No Starch Press. ISBN 978-1-59327-177-0.
- Hughes, L (1998). *Internet e-mail Protocols, Standards and Implementation*. Artech House Publishers. ISBN 0-89006-939-5.
- Johnson, K (2000). *Internet E-mail Protocols: A Developer's Guide*. Addison-Wesley Professional. ISBN 0-201-43288-9.
- Loshin, P (1999). "Essential E-mail Standards: RFCs and Protocols Made Practical". *Programming Internet Mail* (https://archive.org/details/livesofcaptivere00petz). O'Reilly. ISBN 1-56592-479-7.

External links

- "IMAP Protocol Mailing List" (http://www.imapwiki.org/ImapProtocolList).
- RFC 3501 — specification of IMAP version 4 revision 1
- RFC 2683 — IMAP Implementation Suggestions RFC
- RFC 2177 — IMAP4 IDLE command