2

What Is IMAP?

In this chapter:

- *IMAP in a Nutshell*
- *IMAP's Three Interaction Models*
- *Why IMAP?*
- *IMAP and POP: A Comparison*
- *Present and Future of IMAP*
- *Open Source Server Implementations*
- *IMAP-Related Standards and Documents*

This chapter looks at what IMAP is and what distinguishes it from other mail access protocols. We discuss briefly where IMAP is now and where it's headed.

IMAP in a Nutshell

IMAP is a way of accessing electronic mail that is stored on a central server.

Certainly, this statement is true, but there's more to it than that. More precisely, IMAP is a way to retrieve messages from one or more mailboxes on a central server, without ever having to download a single message to local hard disk. The messages remain on the server at all times.

By design, IMAP was intended to provide the same level of functionality for mailbox and message access and management that exists with a mailbox located on a local hard drive. Consequently, IMAP has server operations, such as "search for messages matching such-and-such criteria," that are normally associated with mail clients.

You can see the advantages of IMAP very clearly if you work from several computers (e.g., home computer, office computer, and laptop). With IMAP, you don't have to wonder which computer you were on when you downloaded and read a given message. You know it's still on the server.

With the right IMAP client, you can do all of the following:

- Learn when new messages arrive in any of your mailboxes
- Share your mailboxes with anyone or everyone

- Move messages from one mailbox to another
- Mark messages with flags (such as "Important") that are preserved between IMAP sessions

Another distinguishing feature of IMAP is that, not only is your INBOX stored in a central location, but your *mail folders* (mailboxes, in IMAP parlance) are stored in a central location on the IMAP server as well. As long as you've got IMAP client software,* not only can you read your incoming email from just about anywhere on the Net, but you can access all the mail you archived in your mailboxes, too.

One feature that sets IMAP apart is that it supports not one, but *three* interaction models.

IMAP's Three Interaction Models

IMAP has three models for interacting with your mail: offline, online, and disconnected. The three interaction models are at the core of why, on its own merit, IMAP is the most powerful and flexible mail access protocol. The models are formally defined in RFC 1733 (*Distributed Electronic Mail Models in IMAP4*). Let's take look at the three models.

The Online Model

In the online model, messages are left on the mail server and are manipulated remotely by mail client programs. The mail client maintains an open connection to the server for the duration of the session, that is, until the user decides to end the session.

Whether you prefer the online model has much to do with where you think the higher resource burden should be placed: on the client or on the server. If you believe, as we do, that resources are best centralized when possible, then you're likely to prefer the online model. If you believe that centralized resources should be conserved whenever possible, even at the expense of a greater investment on the part of the user, then you're likely to prefer the offline model. Watch out, though, because placing resource demands on the server requires attention and monitoring of system administrators lest things get out of hand. The online model can put a burden on memory and CPU resources if users tend to keep several mailboxes open, or if mail clients are poorly behaved in some way that results in multiple sessions per user.

---

* Actually, you needn't even have IMAP software on your own machine. You can use a web browser to access your IMAP mailstore; web-based clients are discussed in Chapter 5, *Web-Based IMAP Clients*.

In situations where many users share the same PC (such as a kiosk or computer lab), the online model is the only viable alternative. The same holds true in cases where one user reads mail from many different machines.

An advantage of the online model is that it works splendidly over slow or high-latency communication lines. Let's take IMAP operating in online mode as an example. Early IMAP testing was over a 2,400 bps modem,* and a design goal was that it be no less tolerable than running a host-based program over a dialup connection. In fact, IMAP ended up being more tolerable due to client-based caching of data (such as when the message browser redraws locally instead of over the wire). Even over a slow Internet connection, the online model lets a user manage a shockingly large amount of mail. Suppose a user receives a lot of mail and likes to keep all of it. He might easily have thousands of messages spread across hundreds of mailboxes. When that user dials in to his Internet service provider over a 56 Kbps connection, he definitely does not want to download all his mail in order to read just one particular mail message. Using the online model, he would simply connect to the server and select the message he needed. The only time required would be the time it takes the server to access the message and display a copy of the message in his mail client. Furthermore, with IMAP one can usually select to download an attachment *after* reading the message body.

The Offline Model

The offline model can be compared to a post office box. A person goes to the post office, fetches mail from his box, and leaves the box empty. In the offline model, the mail client fetches messages from the server, stores them on the user's machine, then deletes the messages from the server. The user connects to the mail server, checks for new mail, downloads his new messages, and disconnects. Messages can then be filed into mailboxes or otherwise processed, but such actions are done locally, without the participation of the mail server.

There are advantages of the offline model for both users and system managers. First off, the offline model lessens the amount of space mail consumes on the central mail server. Secondly, for users with a modest amount of mail, it minimizes the amount of time spent connected to the server. The offline model is excellent for the user who always uses the same machine to access his mail and prefers keeping the primary copy of his messages on that machine.

The disadvantages of the offline model are equally compelling. Since all of a user's mail is downloaded to her local computer, she's out of luck if she wants to review her mail archives, unless she happens to be at the location where she downloaded her mail.

---

* Radio was also an early design goal of IMAP. Radio users say that IMAP works wonderfully over radio.

Another disadvantage of the offline model is that it suffers from serious problems of scale for users with significant mail volume, especially if those users have slow Internet connections. Of course, this means that there's a burden on each offline user to either buy more disk storage or store less mail. If a user receives 5 or 10 modest-size mail messages a day, then he might not be inconvenienced by having to download them all to read them. If he gets 100 messages a day with sizable attachments, he's going to wish he'd never heard of email by the time he downloads them all, just to be able to read the first 1 or 2. The offline model can also be expensive in terms of storage on the server. Users who maintain multiple copies of mail in multiple locations generally configure the client not to delete mail from the server after download.

As we'll see later in the discussion of the online model, it's much easier on the user when all he really has to download is an index of his mail.

The Disconnected Model

In the disconnected model, the IMAP client connects to the mail server, synchronizes its list of messages and mailboxes with the server, maybe copies the first few messages into a local cache, then disconnects. In this model, the client queues up tasks and plays them later in a single session.

To the casual user, there should be no difference between the online model and the disconnected model, except that occasionally some operations have a slight latency to them. The disconnected model complements the online model, and in fact, most IMAP clients allow the user to alternate between the two models at will. The intent of the disconnected model is to support users who cannot always connect to a server, but still want to handle mail. Disconnected users are the big winners. The disconnected model allows them to process mail when there's no network available. For example, a user might collect her mail at the airport, board the plane, read and respond to her mail in flight, then reconnect to the server at her destination to synchronize her mailbox and deliver her outgoing mail. There are other situations where disconnected mode comes in handy, too. Take wireless applications as an example—especially those where idle connections rack up per-minute charges. Disconnected mode is made-to-order for connections that charge for time.

It all boils down to picking where you want your bottleneck to be. With the offline model, the desktop machine is the bottleneck. With the online model, the mail server is the bottleneck. With the disconnected model, the network could be the bottleneck. It depends on how fast your server can fork all the processes, or how much bandwidth there is to spare in your network.

Why IMAP?

The short answer is that IMAP solves many of the problems inherent in legacy mail systems. Since the earliest days of networking, the best and the brightest in the computer field have looked for successively better ways to corral individual correspondence into threads of dialog that could easily be used online. This section puts IMAP in the context of several legacy mail models and its competing mail access protocol, POP.

Host-Based Email

Host-based email was the original model for email. If you've been using email for many years, you probably started by using this model. In the host-based email model, the MUA, MTA, and mailstore reside on the same physical machine. Figure 2-1 illustrates the host-based email model.

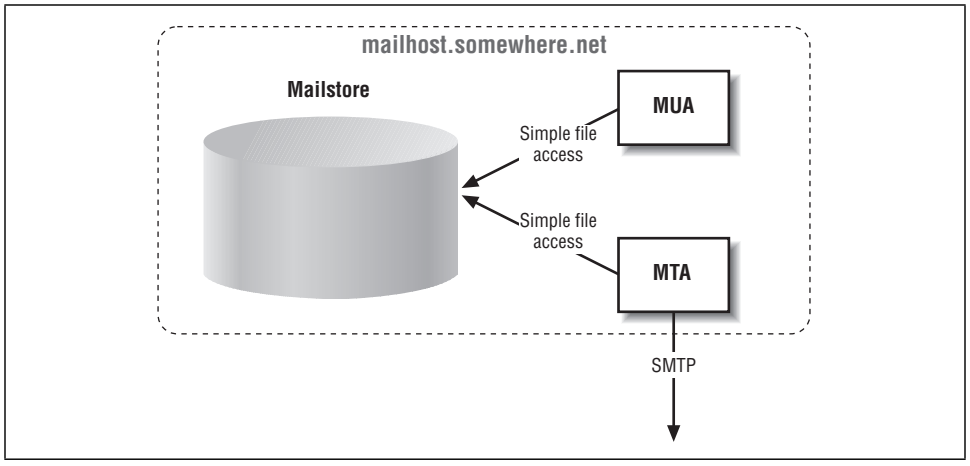

Figure 2-1. Host-based email

Host-based email is a study in single-points-of-failure. Because all functions are carried out by a single system, divisions of responsibility between different parts of the system are unclear. If one part craters, there's a much greater risk that it'll cause the rest of the system to crater. On a host-based email system, the actions of any of the three essential components (MUA, MTA, and mailstore) impact the performance of the others. Host-based email systems are frequently general-purpose compute servers or Internet servers as well. Most enterprise mail systems that have been around for many years began in this model, the shortcomings of which necessitated the development of standardized mail access protocols.

Depending on how you approach it, the host-based email model is either the most or the least secure model around. Some might consider it the most secure because

in a single draconian sweep, you clamp down on the security of the mailstore, transport agent, and user agent in one fell swoop. Of course, this is subject to the Primary Firewall Fallacy: a firewall tricks you into believing that everything on the outside is dirty and everything on the inside is clean. Although host-based email may give you a great deal of control over who can get to your mail jewels, from the outside it provides relatively little protection of your users from one another.

Shared Mailstore

In the fledgling days of PC LANs, more often than not email meant an exalted file-copying program like Novell Mail, Quick Mail, or Word Perfect Office. Those programs copied message files hither and yon between directories on a shared filesystem. Checking for new mail amounted to using a remote file-sharing protocol to see if message files had shown up in a given subdirectory, although the mechanics were usually hidden from the user. The mechanics of scaling systems like that presented profound problems.

Nowadays the acronyms have changed, but the relationship between the user agent machines and the mailstore file server remains essentially the same. In host-based email systems, the child lives at home. In the shared mailstore model, the child has gone off to college, but still gets a stipend from the parents every time he asks for it.

Problems with shared mailstores

The shared mailstore model is fraught with problems, including:

File locking
: With this model, we see the introduction of the bane of Internet mail system designers: the dreaded file-locking problem. The problem goes like this. Suppose you have two pieces of software (the MUA and the MTA), both making changes to the same file (user's INBOX). What happens when they both try to change the file at the same time? That problem is easy to manage on host-based email systems, because under the benign dictatorship of the OS, all relatively well written software can play nicely using standard locking mechanisms. When the mailstore is shared between two hosts, and the hosts lack the governance of such a thing as a network kernel, they will invariably make invalid assumptions about when to write to a file.

  As you start engineering your own enterprise mail system, you are likely to find that any model, even host-based mail, is more desirable than the shared mailstore model.

Synchronization
: When one client makes a change to the mailstore, there's no way for other clients to see that change.

Platform dependence
: No major operating system shares a file-sharing protocol with every other major operating system. As Lincoln might have said, you can share all of the filesystems some of the time, and some of the filesystems all of the time, but you're never going to share all of them all of the time.

Bandwidth usage
: It's clearly inefficient to use a non-client/server database that pulls in the entire database to perform a query. It's equally inefficient to have your mail client pull in your entire mailbox when trying to hunt down a message you received three months ago. Such inefficiencies are multiplied when you pull in your mailbox over the network. That is exactly what happens when you perform an operation on your mailstore, such as a simple search, via a remote file-sharing protocol. It's best to save that kind of bandwidth use for when you truly need to use an entire file, and to instead use client/server mail schemes for routine mail operations.

Mailstore format constraints
: While there are plenty of standards to define what that plaintext looks like (such as ASCII, RFC 822, ANSI, and MIME), there are still subtle differences in the way various platforms choose to store such text. A classic example is the difference between the way Microsoft Windows platforms and Unix platforms end physical lines of text. If you've played the cross-platform game before, you've no doubt pulled up files from your Windows file server in *vi*, only to have them peppered with ^M's. Likewise, you may have pulled up a Unix file in a DOS editor only to find that it has become one long, confusing line of text. Such parochial text-formatting problems go away when you replace the remote file-sharing protocol with a standardized mail access protocol.

Proprietary Mail Schemes

Like it or not, proprietary mail schemes are definitely a significant part of the landscape. In these schemes, one vendor provides the entire soup-to-nuts solution. The MUA, the server, and the gateways necessary to make them talk to anything out in the rest of the world are all provided by one vendor. That vendor decides what features you have. That vendor decides how you scale your system. Effectively, that vendor decides the workflow for your enterprise.

The Faustian thing about these messaging-in-a-box solutions is that they're very tempting for decision-makers to put in place, because they offer a variety of features with deceptively low complexity. They also give the decision-maker a single organization to blame if things go wrong.

Another problem is that proprietary schemes lock you in to a solution. Once you have the scheme in place, your users become reliant on its features. Once your users are reliant on its features, it becomes very difficult to move from the proprietary scheme to a standards-based solution that doesn't have those features.

You also become dependent on the vendor. If you ever have to change vendors (for example, because the vendor's solution doesn't scale to match the growth in your usage), you're stuck. On the other hand, a standards-based solution permits you to change vendors as your needs change.

The real problem is that nobody works in a vacuum. Many proprietary solutions do not rely on standard Internet protocols, but instead depend on a proprietary gateway to talk to the Internet. To accomplish your job, be it commerce, education, or research, you need to be able to communicate via email to people outside your particular enterprise. Besides, open source software increasingly duplicates or surpasses the features and performance of proprietary solutions.

Standardized Mail Access Protocols

In this book, what we mean by a standardized mail protocol is a standardized means of communication between an electronic mail client and server. IMAP is one such protocol. Those protocols, although their implementation sometimes gets a little sticky on this point, are agnostic with regard to platform. That is the core of their value. That is, in fact, the core of the Internet's value. IETF-designed standards have a scope and utility beyond that of any vendor's product. Not to get too preachy here, but the functionality of these standards is the reason why the Internet is a contiguous, mostly highly functional network.

The strategy of using standardized mail access protocols is divide-and-conquer. Internet messaging is divided into tasks that are easily managed by straightforward protocols. The variety of implementation available for each of these protocols gives system administrators much more flexibility in choice when providing services to their end users. The openness of the standard-making process and of the distribution process for most highly regarded Internet software guarantees this.

Use of standardized mail access protocols permits a site to restrict mail spool access to the MDA and the mail access protocol. This, in turn, can reduce or eliminate file-locking problems.

In addition to dividing and conquering, standardized mail access protocols are also good at delegation. Because your IMAP client, for example, can delegate to the IMAP server the task of searching through your mailstore for any messages with a given subject line, the search itself has negligible impact on network bandwidth. The cost of that low impact, however, is generally increased processor use on the mail server.

Standardized protocols free each side from having to worry about the minutiae of having to exchange data with the other. Problems such as character type (ASCII or Unicode), end-of-line discipline, or storage format (unified files or separate data and resource forks) are mostly resolved. Continuing irritants include how best to accommodate various internationalized character sets and how best to encode and decode arbitrary binary files and data streams for encapsulation in a mail message. Various ISO, ANSI, and IETF standards go a great distance in resolving those issues. However, the work is not completely done yet.

All things considered, we hope you'll find, as we have, that use of standardized mail access protocols is the path that holds the most promise and is the least fraught with pitfalls.

IMAP and POP: A Comparison

If you look at which mail access protocols have popular market share, there are only two players on the field: POP and IMAP. POP essentially created the market. It also created a need for a protocol with a great many more features. IMAP filled that need.

POP

POP is the granddaddy of standardized mail access protocols. Although recently you could say that there was more POP client software than IMAP client software, now you can say only that there's more well-implemented POP client software than well-implemented IMAP client software. Therein lies the continuing value of POP: its simplicity. POP is a humble protocol. It doesn't keep track of a variety of message states, it doesn't allow the user to search through her mailbox, and it doesn't even facilitate storing messages in a number of mailboxes. It does one thing and one thing only: it makes the messages available to the user to download to her local machine on demand.

For some users, the limitations of POP are just fine. They use one and only one machine to read their mail, and they download all the messages from their mail server to that one machine and subsequently delete them from the server. In the early days of POP, users wanting to read their mail from more than one machine would have found themselves downloading duplicate copies for their email to each machine from which they wanted to view that mail. At some point, when they thought all the various machines had caught up with their new mail, they finally would delete the mail from the server. Fortunately, though, most modern POP server implementations offer the option of leaving mail on the server after it's downloaded.

For POP users, storing mail in mailboxes means storing mail in mailboxes on the local MUA machine. Searching the mailbox likewise means searching on the client machine. Internet Service Providers (ISPs) for the most part have embraced POP as their mail access protocol of choice because of its modest demands on the mail server. POP is undemanding on the mail server because all the real work is done on your local machine. Mail storage, mail searching, and usually address books are all local data on your workstation.

If you use Internet email at home, work, or maybe a few other locations, as a POP user you'll find yourself with numerous duplicate personal mailstores. Having duplicate copies of your mailstore spread over machines is not necessarily a bad thing. Since with POP your mail is not stored on a server and consequently is not backed up as part of an enterprise-wide backup scheme, having duplicate copies is in essence your insurance against disasters such as hard drive crashes or laptop theft.

It's easy to see that POP users must make more of an investment in processor power and storage to accommodate its shortcomings. POP remains, for many ISPs, the most economical way of providing email service to their users. The days of large-margin low-cost ISP POP services are probably numbered, though. As more users become accustomed to the flexibility and power of IMAP in their own enterprise, they're likely to start demanding the same from their ISP.

For further reading, the current version of the protocol (POP3) is described in RFC 1939.

IMAP

Much of the popularity of IMAP is due, ironically, to the frustrations of longtime POP users. Email for most users amounts to much more than simple day-to-day correspondence. It functions as a database of project history, a scrapbook of sorts of personal relationships, and a database of priorities. If so many of the missions to which we put email feel like a database, why not push some of those functions off to the mail server, where disk, processor and backups, and 24-by-7 support are plentiful? With IMAP, clients not only have the option of operating in offline mode (where mail is downloaded and processed locally), but they may also operate in the much more powerful online and disconnected modes. Online and disconnected modes allow storage and searching to take place on the IMAP server.

As an IMAP user, you suddenly have equal access to your email from any Internet-connected machine running IMAP software.* Your local PC—in fact, every

* If your site offers a web-based IMAP client (Chapter 7, *Installing the Cyrus IMAP Server*), you can access your IMAP mail from any computer running a web browser.

machine you use from day to day—can crash and burn, and your mail won't suffer a bit. One of the first things you're likely to notice is that you no longer need to download all your messages to see what their subjects are. Because of IMAP-MIME integration, you can now download the email message associated with a 50 MB attachment without downloading the 50 MB attachment itself.

So, great! IMAP has everything an email user might want. But what about ISPs, who worry that their profit margin might evaporate with the substantially larger investment in hardware demanded by the load of IMAP? On the server side, implementation details can cut down on disk usage, such as the "single-store" feature (when a message is delivered to more than one user, only one copy of the message is stored on disk). We'd like to suggest that IMAP is more of an opportunity for revenue than anything else is. Instead of admonishing the users for keeping old mail on the mail server, ISPs now have the opportunity to lease the user as much disk space as he likes to hold his mailstore.

IMAP Culture Versus POP Culture

Because of the rich IMAP feature set, the two protocols are more different than they are alike. However, their similarities are worth mentioning. Both POP and IMAP are standard mail access protocols. Remote access in the room next door to the server makes the same demands as remote access from the other side of the world.* In both protocols, the MUA is physically dissociated from the server host.

A key in deciding which protocol is right for your site is your user constituency. POP is an excellent solution for communities of users that always access mail from a single computer. If POP is used in a community where users access mail from more than one computer, then users keep separate and duplicate mail archives on every machine from which they read their mail. POP has been used in environments where all computers running POP clients share a common filesystem, but that approach introduces all the problems with remote file-sharing protocols that we mentioned earlier in the chapter. IMAP archives, on the other hand, can be accessed from any remote platform with IMAP software. If a webmail gateway is in place, change that to any remote platform with a web browser.

The biggest infrastructure difference between POP- and IMAP-based mail systems is the impact of usage patterns on the mail server. IMAP's appetite for server resources is very different from the laissez-faire management of POP mailboxes. In the world of IMAP, both the user's mail archives and her incoming mail are usually stored on the provider's mail server. That server will be hit every time the user reads a mail message, regardless of whether it's a new or an old message. In practice, a POP mailbox is hit by only a single MUA at a time.

---

* Barring the fact that long-distance users may want to encrypt their traffic.

It's not unusual for an IMAP mailbox to be hit by multiple IMAP clients running on different client machines simultaneously. For example, a user might have a program (like *biff*) that periodically polls her mailbox for new mail every five minutes. At the same time, she might leave herself logged in to the IMAP server via PINE 24 hours a day and check her mail from home using Outlook Express every evening. Not to mention the odd stop at the Internet café where she uses Mulberry to check her mail on her coffee break. Such a barrage of IMAP operations against an IMAP mail server's I/O resources should be considered commonplace.

Why Not Both POP and IMAP?

All the discussion of the relative merits of POP and IMAP and how they are appropriate for different sets of users begs the question: why not do both? This is, in fact, a quite reasonable proposition.

If you run a UW or Cyrus IMAP server, you're in luck—both servers come with a POP server that natively accesses the same mailstore as the IMAP server. Another way to do both is to run a POP-to-IMAP proxy (one comes with the UW server). The POP-to-IMAP proxy never talks directly to your mailstore. It simply translates the POP protocol into an IMAP stream, which is then directed at your IMAP server. In fact, if you have a dug-in POP-ulation that you'd like to convert to IMAP, you could just replace your production POP server with a POP-to-IMAP proxy and tell your users, "By the way, we also offer IMAP if you'd like to try it on for size." That is not an altogether uncommon strategy. The downside of using a POP-to-IMAP proxy is that you end up supporting both POP and IMAP clients.

Fortunately, POP and IMAP clients are increasingly becoming the same software—it's just a matter of configuring them differently.

Advantages of IMAP

Here's the rundown on advantages that are unique to IMAP.

Appending to mailboxes

We implied it earlier in this chapter, but it bears repeating. IMAP may be used not only to retrieve messages from your remote mailbox, but also to add them to your remote mailbox. Again, IMAP operates on email much like a database, with the messages being individual records. Using an IMAP client, a user can freely move messages about between his INBOX and additional remote mailboxes of his own creation.

Multiple mailbox support

The first and most obvious demand of an Internet mail access protocol is a mechanism for storing multiple mailboxes on the server. Many email users use filters to sort incoming mail into several different mailboxes. By allowing the user to have multiple mailboxes on the server, IMAP allows users to access their archived mail from any computer, regardless of location.

Remote mailbox management

POP users are used to accessing their messages in a single, contiguous mailbox. Once they're downloaded, messages can be stored in mailboxes in a single MUA installation. In IMAP, the analog of the POP mailbox is a special mailbox called the INBOX. A user can also perform mailbox operations on more than just his INBOX. A typical IMAP user has a number of mailboxes that he's created on the server, each of which he can rename or delete. He can also change certain aspects of the mailbox, such as modifying the mailbox's access control list (if the server so allows) to permit access to the mailbox by other users.

Support for local mailboxes

A user with a given client, while operating in offline mode, may elect to save messages to mailboxes on her local machine. Another user with the same client, operating in online mode, may elect to save his messages to remote mailboxes. Most IMAP clients support both the offline and online modes. The decision of where to save messages is as much a function of user preference, site administrative decisions, and server resource availability as it is a product of protocol features.

Mailbox hierarchies

Most IMAP servers provide for hierarchical mailboxes, in which one or more mailbox names may be grouped together under another name. This encourages users to organize their mailboxes into more than just a flat list. It's perfectly reasonable to have an arrangement of names at the top level (such as "1999," "2000," "2001," "work," "staff," "private"), with mailboxes underneath them (such as the mailboxes "Jan," "Feb," "Mar," and so forth under the name "2000").

Some servers also permit the higher-level names to be a mailbox; for example, both "2000" and "2000/Jan" contain messages. Other servers restrict the higher-level names to be "non-selectable," that is, they are "directories" of mailboxes and not mailboxes in their own right. This varies from server to server and sometimes between different mailstores in the same server.

Remote mailboxes on multiple servers

IMAP clients typically let you manage mailboxes that live on multiple servers. As well as allowing messages to be moved from one mailbox to another on one server, clients often allow messages to be moved freely between different servers.

Persistent mailbox status flags

One aspect of the IMAP mailbox has no direct analog in POP: the message flag. Like file attributes in a directory, message flags hold status information about that message that lasts beyond any particular session. The flags can be changed freely by the IMAP client. A message can be flagged with a standard flag, such as "important" or "answered," or as a "draft" (i.e., a postponed composition). IMAP also provides for user-defined flags, such as such as "Personal," "Work-related," or "This-would-be-good-for-the-IMAP-book."

Server-initiated mailbox status updates

IMAP assumes that you always want to know when you've received new mail or when another IMAP client changes the flags of a message or removes a message from the mailbox. Any time that you perform an operation on a mailbox, the server can add to its response to the client, "Oh, by the way, you now have N messages" or "Oh, by the way, message M now has such-and-such flags set." In more restricted circumstances, the server can also add, "Oh, by the way, message M has been removed."

Companion configuration protocols

One of two protocols is typically employed for remote storage and location-independent access of IMAP client configuration options: IMSP or ACAP. IMSP (Internet Message Support Protocol), the older of the two, is designed exclusively to store IMAP client configuration and personal address books. ACAP (Application Configuration Access Protocol), the new kid on the block, is designed for storage of Internet application data and configuration. Not only can it store the client configuration and address books, but also other application data, such as bookmarks for your web browser. ACAP's flexibility makes it attractive to use for IMAP, and it can be pressed into service for other missions as well. IMAP and ACAP are discussed in more detail in Chapter 17, *Remote Configuration Storage*.

IMAP extensions

If IMAP had no other features save for this one, it would still have the framework necessary to provide a robust and popular mail access protocol. That framework permits clients to request a list of capabilities of each IMAP server with which it

talks. It also ensures that the service life of IMAP will extend far beyond that of POP, because obsolete features can be abandoned and new features can be created on demand.

Performance advantages

IMAP provides a handful of ways to cut down on the amount of time the end user must spend managing her email. Perhaps the most frequent time saver is server-based searching. With IMAP, the performance of a search depends on server resources, not on the communication bandwidth and resources of the client.

Another reason the user experience is streamlined is that the IMAP server sees each message not as a single RFC 822 blob, but as a collection of headers followed by a collection of MIME body parts. Each collection can be listed like a directory, giving the user the ability to download each MIME component as he sees fit. That means that when the user receives an email message in the middle of his workday from Uncle Kevin with 12 MB of digital photos attached, he can read the message and wait until later to download the attachments. The ability to download the message structure independent of the message itself means that IMAP is the traveler's friend. Although John Doe may have a speedy Internet connection at work, the best he can expect while he's on the road is a 56 Kbps modem connection.

IMAP supports non-email data

IMAP can be applied to Usenet News or bulletin board discussions. Many IMAP servers are non-discriminating about where they get their data. That data need not be a product of IMAP or SMTP delivery. Any RFC 822/MIME-compliant message will do.

Shared mailboxes

One of the most productive features IMAP brings to the table is the ability to share read-write access to any given mailbox. This means that the mailbox that receives mail for *help@yourenterprise.net* can be opened by each person in your call center at once, and everyone will be able to keep track dynamically of which messages have been opened.

Another popular use for shared mailboxes is for global announcements. Announcements such as "A huge chlorine cloud is descending on campus" can be added to an enterprise-wide announcements mailbox so that users can read it at their leisure. This feature is frequently combined with IMSP or ACAP storage of client configuration, so users may be subscribed to the announcement mailbox by the administrator.

Mailbox sharing also helps reduce another bane of system administrators: account sharing. If users are able to share mail and other data by just changing an access control list on a mailbox, they have one less reason to give out their password to others to share data.

Finally, mailboxes may also be shared anonymously. Several Internet mailing lists' archives are shared in this manner, providing IMAP users with built-in forwarding, new message notification, and searching capability usually not found in mailing list archives.

Feature Breakdown

Table 2-1 compares POP's and IMAP's features.

Table 2-1. POP and IMAP Feature Inventory

Feature	IMAP	POP
Primary operational model	Online/ disconnected	Offline/ disconnected
Open Internet standard protocol	✓	✓
Free/open source server implementation available	✓	✓
Clients available for a variety of platforms including Mac, PC, and Unix	✓	✓
Supports all three models: online, disconnected, and offline	✓	✗
Natively supports access to a single mailbox from multiple computers	✓	✗
Supports interactive access to multiple mailboxes on different servers	✓	✗
Supports concurrent read/write access to shared mailboxes	✓	✗
Supports access to data other than email messages (e.g., NNTP)	✓	✗
Appending to mailboxes: supports storage, not just retrieval, of messages	✓	✗
Persistent message flags (e.g., "important," "seen," or "answered")	✓	✗
Persistent message IDs	✓	✓
Minimizes server storage	✗	✓
Minimizes connect time	✗	✓
Sends outgoing messages via SMTP	✓	✓

Present and Future of IMAP

In the not-so-recent past, IMAP was largely thought of as an overly complex method for managing email, and POP was considered just fine. However, nearly everyone supports or promises to support IMAP these days. That makes the size of the development investment in IMAP possibly second only to HTTP. You would think this would be a good thing. As you'll read later, however, at least on the MUA side of the house, there's some sloppy implementation going on.

This actually ends up being a *good* thing. In an open-standard marketplace, the more users who reject the one or two large-market-share apps in preference to the plethora of smaller-market apps, the more diversity there is in the market. Some of the best IMAP implementation going on right now is from companies you probably have never heard of before. If the one or two market leaders hadn't given away their market share, the IMAP market would be as locked up as the PC operating system market.

With the added complexity of IMAP comes greater flexibility. With flexibility comes a broader base of technologies on which it can be implemented, ranging from interactive voice response to wireless personal digital assistants (PDAs). In one sense, the success of a standard might be measured by the number of related standards it inspires.

The glory days of IMAP are just beginning. Here's a brief list of where we stand as of this writing:

- Dozens of IMAP implementations exist for nearly every desktop and server platform.
- IMAP is being used as a backend for web-based email interfaces, further adding to its ubiquity.
- The IMAP client market for PDAs, notably Palm OS and Windows CE, is just starting to take off, with a handful of clients for each.
- While users may not be storming the ISP Bastille asking for IMAP by name, the feature list they are demanding fits IMAP to a tee.
- Commercial Internet messaging products now incorporate IMAP support as a core feature.
- University messaging environments now regard IMAP as standard operating procedure.
- IMAP is starting to be offered on architectures like cellular phones, but the jury is still out on how useful IMAP will be in those environments.

Many of the IMAP users in the future will probably have no idea that IMAP is what makes their extended messaging environment possible. Here's where IMAP is headed:

- As universal multi-device connectivity becomes more common, IMAP will surely play a central role in coordinating simultaneous access to users' mailboxes from many devices at one time. POP simply doesn't scale in that regard.
- An unexpected wrinkle in the story of IMAP has been its growth as a back-office protocol supporting web-based email frontends (see Chapter 5, *Web-Based IMAP Clients*). IMAP's importance will probably be equally split between back office support of web-based frontends and direct interaction with client user interfaces.
- If the activity surrounding the development of extensions to IMAP is any indication, we can look forward to IMAP having a productive life, far exceeding that of POP.
- Finally, as the Internet moves into an age when ever fewer users need to know that such things as IP addresses, netmasks, and mail access protocols exist, IMAP's destiny may be that of the "man behind the curtain."*

Open Source Server Implementations

For nearly a decade, there have been only two appreciable open source IMAP servers: the University of Washington's IMAP server and Carnegie Mellon University's Cyrus IMAP server. Both are robust, time- and user-tested servers with an install base to rival any commercial alternative.

We will go into more detail in later chapters about the interesting histories and specific features of each server, but let's take a brief look at them for the time being.

University of Washington IMAP Server

The University of Washington server is the reference implementation of IMAP. It was written by Mark Crispin, the inventor of IMAP. It was started in 1988 at Stanford University as a C rewrite of the original Interlisp client and DEC-20 assembly language server. When Crispin changed jobs for the University of Washington late in 1988, the IMAP project went with him.

The University of Washington IMAP server strives for compatibility with existing Unix systems. If you've stored your mail in a given format on a Unix system over

* A reference to *The Wizard of Oz*.

the course of the past few decades, chances are that the UW server can read that format out of the box.

The UW server has a number of interesting experimental extensions, such as multiple append and server-based sorting and threading. The UW server also supports an extensive list of international character sets.

The UW server is very modular—it is easy to add support for another mailbox format or SASL authenticator by writing a code module and relinking.

Carnegie Mellon University Cyrus IMAP Server

The CMU Cyrus Server is a component of a project called Project Cyrus.* Started in 1994, the Cyrus project was started because the management overhead of running the existing proprietary system was getting to be too high. CMU was unable to keep up with client development in particular and wanted to use commercial off-the-shelf (COTS) mail clients. Project Cyrus was created to provide a next-generation messaging system that relied heavily on Internet standards and was highly scalable as well as modular, supported disconnected mode, and enabled freedom from legacy architecture. Cyrus was a rejection of the idea of basing an email system on software that just copied files around from place to place in a file-system. Historically, the lack of reliability of such systems has only been exceeded by their lack of scalability.†

Although the core of the Cyrus project is the Cyrus IMAP server, CMU has developed the related protocols IMSP and ACAP, and implementations of those protocols to support their IMAP server, as well. The IMSP and ACAP servers offer a way to store the user's personal client settings remotely. The SASL library provides a way for any Internet standard-based application (client or server) to perform Internet standard authentication. CMU's Cyrus IMAP server also implements the SIEVE server-side filtering language, to boot.

Perhaps its greatest quality is the Cyrus server's feature richness. Given the right client, users benefit from such features as support for IMAP quotas and mailbox access control lists. The University of Washington server is rich with support for differing types of email storage formats. If you've stored your email in a given format on a Unix system over the course of the past few decades, chances are that the UW server can read that format.

The difference between the UW and Cyrus servers can be summarized as follows: UW is a generalist and Cyrus is a specialist.

---

* *http://asg.web.cmu.edu/cyrus/.*

† Not that people learn, however. Many popular LAN-based proprietary email systems are, under the hood, little more than file-copying utilities. Internet email standards rule.

The UW server is engineered to be completely compatible with Unix mail. It assumes the worst case: a site where the mail server also acts as an interactive login server for the general population and where users also access their mail directly with host-based mail clients. The UW server does not add any management overhead—all quota and access control is handled by the kernel (e.g., with Unix file and directory permissions).

Cyrus is engineered to run on so-called "black box" servers, where the user's mail is read by no software other than the Cyrus server. Cyrus offers rich support for IMAP quotas and mailbox access control lists and a great deal of IMAP-specific management control. Cyrus also obtains considerable performance benefit from not having to be compatible with ancient email software.

Midrange-to-large IMAP sites that choose UW generally modify it in various ways to make it more of a specialist, like Cyrus. A number of large IMAP sites do use UW; usually these are large sites that don't want the management overhead of Cyrus.

IMAP-Related Standards and Documents

All the standards and related documents in this section are Request For Comment (RFCs) documents. There is a plethora of IMAP-related Internet Drafts (would-be standards that are still working their way through the approval process), but they change so rapidly that they won't be mentioned here. Everyone has favorite RFC archives. Here are ours. Each of the sites lets you choose the geographically closest archive from which to retrieve your documents:

- Internet RFC documents (*http://www.nexor.com/info/rfc/index/rfc.htm*)
- Internet Drafts (*http://www.nexor.com/info/internet-drafts/id.html*)

A word about these documents—they're the epitome of "hit the ground running" docs. They're meant to be terse and narrow, much like their Unix manpage cousins. Despite their laissez-faire name, RFCs are the canonical standards documents of the Internet. If you have a bet with someone and need an indisputable source to settle the argument, turn to the RFC.

Table 2-2 is a snapshot of the current RFC standards related to IMAP. For a more comprehensive list, you can do a database search on "IMAP" at one of the previously mentioned URLs. In Table 2-2, the most important document is the *Internet Message Access Protocol Version 4rev1*, RFC 2060, by Mark Crispin. Consider it the defining document of the core features of IMAP. Second behind it would be RFC 2683 (*IMAP4 Implementation Recommendations*), which is necessary to read to understand the IMAP "folklore."

Table 2-2. IMAP-Related RFCs and Internet Drafts

RFC Number	Authors	Title
1731	J. Myers	IMAP4 Authentication Mechanisms
1732	M. Crispin	IMAP4 Compatibility with IMAP2 and IMAP2bis
1733	M. Crispin	Distributed Electronic Mail Models in IMAP4
2060	M. Crispin	Internet Message Access Protocol Version 4rev1
2061	M. Crispin	IMAP4 Compatibility with IMAP2bis
2086	J. Myers	IMAP4 ACL extension
2087	J. Myers	IMAP4 QUOTA extension
2088	J. Myers	IMAP4 non-synchronizing literals
2095	J. Klensin, R. Catoe, P. Krumviede	IMAP/POP AUTHorize Extension for Simple Challenge/Response
2177	B. Leiba	IMAP4 IDLE command
2180	M. Gahrns	IMAP4 Multi-Accessed Mailbox Practice
2192	C. Newman	IMAP URL Scheme
2193	M. Gahrns	IMAP4 Mailbox Referrals
2195	J. Klensin, R. Catoe, P. Krumviede	IMAP/POP AUTHorize Extension for Simple Challenge/Response
2221	M. Gahrns	IMAP4 Login Referrals
2342	M. Gahrns, C. Newman	IMAP4 Namespace
2359	J. Myers	IMAP4 UIDPLUS extension
2595	C. Newman	Using TLS with IMAP, POP3, and ACAP
2683	B. Leiba	IMAP4 Implementation Recommendations