In this chapter:

- *Why Store Client Configurations on a Server?*
- *IMSP, ACAP, or LDAP?*
- *IMSP*
- *ACAP*

17

Remote Configuration Storage

Two protocols that complement IMAP, both with roots in the academic world, are the Application Configuration Access Protocol (ACAP) and the Internet Message Support Protocol (IMSP). Both protocols are used for storing user preference and address book information on a remote server, encouraging Internet desktop ubiquity.

Web sites that were once search engines are now calling themselves "portals." What makes a search engine a portal is that what used to present the same resources to all users is now personalized with customized email, scheduling, remote bookmarks, and address book functions.

Remote configuration is all part of a push to make the Internet less tied to a specific physical location, whether to your desk at work, your PC at home, or the notebook in your briefcase. As personalization spreads to more Internet applications, the ponderousness of Internet software and operating systems should decrease. The piece that makes thin client computing work is the ability to do anything anywhere. Remote configuration and personalization make that possible.

Why Store Client Configurations on a Server?

Your site may benefit from centralized configuration management. It certainly makes sense for users who want to access their email from more than one place, especially those who use shared computers. Finally, because we've all learned the hard lessons of the total cost of ownership for the PC on the desktop, centralized configuration management makes fiscal sense, too.

Benefits of Centralized Configuration Storage

The lighter Internet applications become, the more likely they are to depend on having preferences stored in a central location. Here are some examples of environments that benefit from centralized user preference management:

General access computer labs
: Users who read email in general access computer labs want the same email preferences regardless of which workstation they happen to be using on a given day. Those users probably don't want the hassle of carrying their email preferences around on a floppy disk.

Mobile computing environments
: Mobile users have more than one computer, usually an office computer, a home computer, and a laptop. Sometimes the laptop is checked out at random from a pool of laptops. Mobile users don't have the time to change preferences every time they change computers, nor do they want to carry the preferences around on a disk and risk losing the disk.

In-house computer hardware shops and value-added resellers
: Shops that distribute large numbers of machines could set up machines to grab a generic email client configuration over the network. That configuration would customize the email client for the site, cutting down on the amount of work required on the part of the end user. DHCP is used in the same way, permitting many sites to release end users from having to configure their machine for a particular set of IP parameters. ACAP and IMSP permit users to instantly become familiar and comfortable with a possibly unfamiliar MUA.

Environments that have a centralized backup strategy
: By storing user preferences in a central location, it's easy to back up and restore your users' preferences without any complex changes to your infrastructure.

Environments that support use of PDAs
: Users often want the same email options on their PDAs (Personal Digital Assistants) as on their full-size computers. PDAs and sub-notebooks usually have no floppy drive, and alternatives to floppies (such as flash ROM) are very small (easy to lose) and expensive. Such devices do have the ability to connect to the network and download preferences from an Internet-accessible database. The ability to download preferences is also a very attractive option, given the limited local storage on PDAs.

Essentially, remote profile storage with IMSP or ACAP adds even more value to the enterprise infrastructure than it does to an individual's productivity. The amount of individual effort saved by central profile storage is a convenience at best. The

amount of effort saved by central IT support facilities is potentially huge. As the amount of hands-on support required for each machine diminishes, so does the total cost of ownership.

Much to the amusement of longtime central computing types, the desktop computer has proved to be one of the most expensive means of getting computing to the people. Although the entry cost might be low, the sluggishness of some desktop operating systems has exponentially increased the amount of legwork and end-user machine customization necessary to support each application, as well as baseline system stability.

IMSP, ACAP, or LDAP?

If centralized user preference management were thought of as a nail, not everyone would agree what kind of hammer is needed. Two come to mind: IMSP and ACAP. One doesn't: LDAP.

There are a couple of reasons why LDAP doesn't really fit here. The lesser of the two is that LDAP is tuned to perform best in a read-intensive environment. ACAP, on the other hand, is designed to work in a mixed read/write environment, such as one where users are constantly updating preferences.

But the greater of the two reasons is that the structure of a user's configuration information, such as bookmarks and addresses, is defined by the administrator under LDAP and is static. Under ACAP, the ordering is determined by the user and can be changed on the fly. LDAP is a good fit for enterprise-wide information; it's not necessarily a good fit for email preferences storage, because each user tends to cultivate a very personalized directory of their own.

Users are likely to use LDAP-based services for tasks like finding someone's email address. Once they find it, however, they're likely to want to store it in their own address book, so they can make it meaningful in ways that are not supported in LDAP, such as:

- Tying the address to an easier-to-type nickname
- Taking advantage of auto-completion in their client
- Adding additional information that wouldn't necessarily be germane to a central directory, such as notes from the last meeting, private telephone numbers, names of spouse and children, etc.

Briefly put, ACAP is remote configuration management for the people. Table 17-1 is a brief comparison of the features of IMSP, ACAP, and LDAP. The table boils down many of the issues central to choosing a user preference management protocol.

LDAP is weak in the area of per-user attribute storage and client-defined attributes, but IMSP and ACAP are strong in this regard. With its server-side searching and large datasets, ACAP seems much more scalable and flexible than its earlier counterpart, IMSP.

Table 17-1. Comparison of IMSP, ACAP, and LDAP

Feature	IMSP	ACAP	LDAP
Optimized for read and write performance	✓	✓	✗
Data can be written by the client	✓	✓	✓
Server-side searching	✗	✓	✓
Supports large datasets	✗	✓	✓
Supports disconnected use	✓	✓	✗[a]
Supports client-defined attributes	✓	✓	✗
Open standard (non-proprietary)	✓	✓	✓
Supports ACLs (access control lists)	✓	✓	✓
Supports per-user authentication	✓	✓	✓
Supports per-user storage of information	✓	✓	✓
Supports hierarchical organization of information by the user	✗	✓	✗

[a] LDAP doesn't support disconnected mode in the sense that we're using in this book, but LDAP does nicely support the multitier equivalent of disconnected mode, where subsets of a directory/database are delegated to second- or third-tier LDAP servers. If those servers become disconnected from the top-level LDAP server, they queue up their transactions until the connection is restored.

Keep in mind also that, with traditional directory services, the protocols are designed to give the administrator a great amount of control. In IMSP and ACAP, the data is usually owned by the user, and the protocol is designed to give the control to the user. Users have the freedom not only to edit their data directly via their client, but to create hierarchies and move data around in the hierarchy, much the way they do when the data is stored on the local hard drive.

To quote Matt Wall, former manager of Project Cyrus and co-founder of Cyrusoft, Inc., on IMSP and ACAP: "We believe in the concept of 'the right tool for the right job'. We have no love for reinventing the wheel, but in researching the available options in the context of Project Cyrus, we discovered this particular type of precision screwdriver. Trying to get one of these other protocols [to work for remote preference storage] is like using a heavy-duty hammer or a wrench to get this particular screw attached."

Having narrowed our choices down to IMSP and ACAP, let's look at each in more detail.

IMSP

IMSP is an Internet protocol that allows application programs to store program options and user information—such as personal and shared address books—on a remote network server. IMSP thus provides retrieval of client configuration information, which is traditionally stored on local disk, from anywhere on the network.

The original IMSP specification was written by members of the Project Cyrus team at CMU. The first IMSP server was released in 1994, also by CMU. Development of IMSP ceased in 1995, when it became evident that there was a need for a protocol to store client preferences that applied to other types of Internet applications, not just email. At that point, IMSP was reengineered and renamed Application Configuration Access Protocol (ACAP). ACAP is discussed later in this chapter. IMSP never made it into the standards track; its status is that of "experimental draft." Despite its experimental status, IMSP is still alive and well, in use in production environments at over 1,000 sites, with more than one million end users. Because there are still very few ACAP-capable MUAs and ACAP servers, IMSP is still a good option for email client configuration storage. There are both stable IMSP clients and servers, and the protocol has been proven to work.

IMSP Specification

The IMSP Internet Draft is available from CMU at *http://asg.web.cmu.edu/cyrus/rfc/imsp.html*.

Cyrus IMSP Server

The recommended IMSP server is the CMU's freely available Cyrus IMSP server, Version 1.6a1. Despite the alpha release number, that release of the server has been tested over time and proven stable.

Where to get IMSP

The location of the server source distribution is *ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/cyrus-imspd-v1.5a6.tar.gz*.

How to install and configure IMSP

In an appropriate directory, unpack the source distribution:

```
% zcat cyrus-imspd-v1.5a6.tar.gz | tar xvf -
```

Next, compile the sources and install the software. IMSP installs under */usr/local* by default. There are configuration options, such as authentication method, that you may want to specify explicitly. To see the possible options, run *configure --help*:

```
% ./configure -with-login=unix_shadow
% make all
# make install
```

Once IMSP is built and installed, there is some post-installation configuration you'll need to complete. First, create a directory where IMSP will store its data files:

```
# mkdir /var/imsp
```

Next, edit */etc/services* and add an entry for IMSP. The line in */etc/services* should look like this:

```
imsp 406/tcp        # Internet Message Support Protocol
```

IMSP has a single global configuration file (*/var/imsp/options*) and an individual user options file for each user. User options are stored under */var/imsp/user/username/*.

Set up a global IMSP *options* file. As a start, copy the *options.sample* that is provided with the IMSP distribution to the data directory:

```
# cp options.sample /var/imsp/options
```

IMSP options are specified as attribute-value pairs with optional flags: *attribute [flag] value*. Possible flags include:

R Read-only attribute

N Read-only attribute; invisible to users, and used for options that pertain to administrators only

W Writable attribute; the value can be changed by the client and saved in the user's personal options file

Add the following line to allow users to automatically create a new personal options directory, if the directory does not already exist:

```
imsp.create.new.users N +
```

The following lines are other entries that are commonly found in the global IMSP options file:

```
common.date R                          Current date and time
common.delivery.hosts R smtp.blah.edu  Local SMTP host
common.sent.mailbox R (INBOX.sentmail) Default sent mail folder name
common.domain R yourdomain.edu         Local mail domain
imsp.admin.all N (johndoe)             IMSP administrative user
imsp.user.quota r 2048                 Quota on user options
```

With all that done, you're now ready to run IMSP.

How to run IMSP

The Cyrus IMSP server runs as a standalone daemon and listens for requests on port 406. To start the IMSP server, type the command:

```
# ./imspd
```

To test IMSP, telnet to the IMSP port and type in the commands shown in the following example session. If you get the response OK at startup and on login, then everything's working fine:

```
% telnet localhost imsp
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* OK Cyrus IMSP version 1.5a6 ready
001 login johndoe xxxxxxxx
001 OK user johndoe logged in
001 logout
```

Getting help

The Cyrus IMSP server is supported on the *info-cyrus* mailing list. To subscribe to the list, send a message with the text "subscribe info-cyrus" to *info-cyrus-request@andrew.cmu.edu*. An archive of the *info-cyrus* list is also available, and it's a good idea to check the archive before posting questions to the list. To view the archive, point your IMAP client to *cyrus.andrew.cmu.edu*. Use anonymous as the username and your email address as your password. The name of the folder is *archive.info-cyrus*.

Cyrusoft's "The Cyrusoft Guide to IMSP" is useful to refer to when configuring IMSP options, and it has information on how to serve up global address books via IMSP. To retrieve a copy of the Guide, visit *http://www.cyrusoft.com/support/faq/mulbpapers.html*.

IMSP Clients

Two IMAP clients support IMSP: Cyrusoft's *Mulberry* and MessagingDirect's *Execmail*. Cyrusoft's *SilkyMail*, a webmail IMAP client, also supports IMSP. IMSP support is planned for a future release of MessagingDirect's Webmail client: *Execmail Web*. In fact, the combined popularity of IMSP and web-based email has driven the development of an IMSP client library to be released soon as part of the PHP distribution.*

* PHP (*http://www.php.net/*) is a server-side scripting language for creating dynamic web pages and is used in several popular web-based IMAP clients.

ACAP

ACAP is an Internet protocol used by client programs to store and retrieve client program information, such as bookmarks, address books, and program preferences. ACAP provides more than just access to preferences from many locations; it can also provide access from any Internet application—not just email clients.

Like IMSP, ACAP is *not* a directory service, but rather, a protocol with a different purpose. ACAP is intended to work in harmony with directory services, not in competition with them. ACAP fills the niche between a directory service, like LDAP, and a limited-service support protocol like IMSP. ACAP, in fact, offers some specialized functions that directory services do not support:

Remote storage of email account data
: It's becoming more and more common for Internet email users to have more than one mail account (e.g., an account at work and an account at home on an ISP). Users access multiple accounts from the same machine and/or access the same accounts from different machines. They may also use more than one program that requires email account configuration information. ACAP supports the storage of email account data.

Remote storage of bookmarks
: Storing bookmark URLs is common in Internet applications such as web browser and FTP clients. Users need to access the same bookmarks from different client programs and from different machines. ACAP supports synchronization of bookmarks between multiple applications and systems, and even allows sharing a single bookmarks list between users.

Remote storage of roles
: It has become common for Internet mail users to receive and compose email in the capacity of different roles, or "personalities." For example, a user might use one email personality at work to communicate with colleagues and a different identity at home to communicate with friends and family. ACAP provides a way to store email composition preferences.*

Remote storage of a common MOTD
: ACAP supports remote storage and access of a "Message of the Day" greeting, used by system administrators to communicate important information to all users when they begin to use a system. This is particularly useful to system administrators who manage black box systems where users do not log in directly to a shell account, and thus do not see the traditional Unix "motd."

* We've seen a case where LDAP was used to support several "roles" per user, and the workaround (or more accurately, "kludge") was to assign each user a username to go with each identity.

Remote storage of the mailboxes dataset

This use lets you separate the information about the mailboxes from the mailboxes themselves. Clients and servers that used ACAP in this manner would keep lists of mailboxes, their new message and other status, and even information about which server on which they exist in a dedicated database, wholly separate from the mailboxes themselves.

ACAP, part of CMU's Project Cyrus initiative, evolved from IMSP. IMSP is very successful in its purpose, which is to store limited email client configuration options. It became evident once IMSP achieved widespread use, however, that there was a need to apply the underlying fundamentals of IMSP to other types of Internet clients, rather than just email. More generalized Internet preference storage was an idea whose time had come. In January 1998, the ACAP RFC was published, followed quickly by the initial release of the Cyrus ACAP server two months later.

Like IMAP itself, anyone implementing ACAP will find it's quite a moving target. The Cyrus ACAP server, for example, is the grand dame of ACAP servers, and has undergone a multiplicity of face-lifts since its initial release.

ACAP Specification

The ACAP specification is available from the IETF in RFC 2244 *(http://ietf.org/rfc/rfc2244.txt)*.

ACAP-related RFCs and drafts

The strength of a good Internet protocol lies not only it its evolution as a standard, defined in an RFC, but also in its extensibility. Like Telnet, IMAP, and SNMP MIBs, ACAP is a popular standard for which to write and implement useful extensions. Here's an overview of some of the more important ones.

The ACAP Dataset Model Internet Draft (*http://ietf.org/internet-drafts/draft-ietf-acap-dataset-model-01.txt*), primarily intended for developers of ACAP clients, provides guidelines on how to design and access ACAP datasets and explains the relationship between ACAP attributes, entries, datasets, and dataset classes.

The ACAP Bookmarks Dataset Class Internet Draft (*http://ietf.org/internet-drafts/draft-ietf-acap-book-02.txt*) defines a standard ACAP dataset class for storing bookmark URLs.

The ACAP Email Account Dataset Class Internet Draft (*http://ietf.org/internet-drafts/draft-ietf-acap-email-02.txt*) defines a standard ACAP dataset class for email accounts and a common option for indicating a default email account.

The ACAP Email Personality Dataset Class Internet Draft (*http://ietf.org/internet-drafts/draft-ietf-acap-pers-02.txt*) defines a standard ACAP dataset class for

outgoing email identities (also known as roles or personalities) and a common option for indicating a default.

The ACAP Message of the Day Dataset Class (*http://ietf.org/internet-drafts/draft-ietf-acap-motd-dataset-00.txt*) describes a common format for storing MOTD information in ACAP. It explains how site administrators may configure their ACAP MOTD service to allow multiple groups within the site to provide custom MOTD information and how a client should access and use this information.

Cyrus ACAP Server

The CMU developed the first, and the first free, ACAP server. It continues to be the benchmark ACAP server and is a great choice for those who want to dive into ACAP.

Where to get ACAP

The latest version of CMU's Cyrus ACAP server is available at *ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/*. At the time of this writing, the latest version is Version 0.3. Cyrus ACAP is available in both source and binary (Solaris and Linux) distributions.

Cyrus ACAP 0.3 has several dependencies:

CMU SASL
: CMU *acapd* requires CMU's SASL library (*ftp://ftp.andrew.cmu.edu/pub/cyrus-mail/cyrus-sasl-1.5.5.tar.gz*) for authentication.

SML/NJ
: The backend of the Cyrus ACAP server is written in SML, a high-level programming language. SML/NJ (Standard ML/New Jersey implementation) is Bell Labs' SML compiler. Version 110.0.6 of SML/NJ is recommended. SML/NJ is available at *ftp://ftp.research.bell-labs.com/dist/smlnj/release/110/*. Note that if you're installing a binary distribution of Cyrus ACAP, you don't have to install SML/NJ.

GNU Make
: The ACAP documentation recommends using GNU Make (*ftp://ftp.gnu.org/gnu/make/*).

How to install and configure ACAP

The instructions outlined here are for installation from the source distribution.

You should have GNU Make, SML/NJ, and SASL installed on your system before installing ACAP. Make sure the *sml* binary is in your path so that the configure script will detect it. Unpack the ACAP source distribution.

```
% zcat cyrus-sml-acapd-0.3.tar.gz | tar xvf -
```

Run the *configure* script, *make*, and *make install* to install *acapd* on your system.* The default install prefix is */usr/local/*.

```
% ./configure
% make
# make install
```

If ACAP is not defined in your */etc/services* file, then add the following line:

```
acap 674/tcp       # Application Configuration Access Protocol
```

Add the following line to */etc/inetd.conf*, then restart *inetd*:

```
acap stream tcp nowait root /usr/cyrus/bin/frontend frontend
```

Create the ACAP directories:

```
# mkdir /var/acap
# mkdir /var/spool/acap
```

Then finally, start the backend ACAP process:

```
# cd backend
# backend-acapd &
```

How to use ACAP

The Cyrus ACAP server runs as a standalone daemon and listens for requests on port 674. To start the ACAP server, type the command:

```
# cd backend
# backend-acapd &
```

To test ACAP, *telnet* to the ACAP port and type in a few commands, as shown in the following example session. If you get the response `*` at startup, then everything's working fine:

```
% telnet localhost acap
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
* Acap (Implementation "SML Frontend, Carnegie Mellon Project Cyrus") (Context
Limit "100") (Sasl "PLAIN" "ANONYMOUS")
0 AUTHENTICATE "ANONYMOUS" "johndoe"
0 Ok "Welcome"
1 LOGOUT
* BYE "have a nice day"
1 OK "LOGOUT completed"
Connection closed by foreign host.
```

* We found an error in the Makefile. After running configure, you may need to add the following line to the beginning of the "install" target: `@for d in $(SUBDIRS); \`.

Where to get help

The *info-cyrus* mailing list is the best place to go for technical questions about the Cyrus ACAP server. To subscribe to the list, send a message with the text "subscribe info-cyrus" to *info-cyrus-request@andrew.cmu.edu*. An archive of the *info-cyrus* list is also available, and it's a good idea to check the archive before posting questions to the list. To view the archive, point your IMAP client to *cyrus.andrew.cmu.edu*. Use "anonymous" as the username and your email address as your password. The name of the folder is *archive.info-cyrus*.

ACAP Clients

The following clients currently include support for ACAP:

- Cyrusoft's Mulberry
- MessagingDirect's Execmail 5.0
- Qualcomm's Eudora Pro

University of Washington plans to add ACAP support to a 4.x release of the popular PINE email client, but it had not been added as of Version 4.20.