NAC POSITION PAPER

INTEROPERABILITY: ELECTRONIC MAIL SYSTEMS

ABOUT NAC

The Network Applications Consortium (NAC) is a strategic organization dedicated to increasing the availability and quality of applications for enterprise-wide computing by:

- publishing papers that state NAC's strategic vision of the industry's direction;
- educating vendors about enterprise-wide computing requirements;
- promoting, facilitating, and documenting collaboration among NAC members;
- advising distributed computing vendors on corporate planning and product development policies.

With these goals in mind, NAC is publishing *Interoperability: Electronic Mail* to help foster productive discussion among vendors and users. The paper discusses the issues related to interoperability among electronic mail systems in large, heterogeneous enterprises, such as those in use by NAC member companies. Members include:

ABB Power T&D Co.
Arizona Public Service Company
Banyan Systems Inc.
Carolina Power & Light Co.
Conner Peripherals
Intel Corporation
Martin Marietta Energy Systems
NYNEX
Pacific Gas & Electric
Public Service Electric & Gas
Sprint Corporation
Tektronix, Inc.
University of Michigan
American Bureau of Shipping
Australian Bureau of Statistics
Bell Atlantic Mobile Systems, Inc.
Compaq Computer Corporation
Continental Grain Company
International Finance Corp., World Bank
MCI Telecommunications
Nike, Inc.
Pennsylvania Blue Shield
Resolution Trust Corporation
St. Jude Children's Research Hospital
Texaco
United States Marine Corps

This paper is the result of NAC's Strategic Interest Group (SIG) process, a collaborative effort involving a subset of NAC members whose mission is to provide a cohesive NAC viewpoint on a particular industry sector or technical topic. The following NAC members were instrumental in preparing this position paper:

Continental Grain Company — *Eric Dickstein*
Continental Grain Company — *Hilly Fuchs*
Continental Grain Company — *Bill Klauk*
Martin Marietta Energy Systems — *George Farris*
MCI Telecommunications — *Brian Plackis*
NYNEX — *Doug Savary*
Pacific Gas & Electric — *Art Beckman*
Pacific Gas & Electric — *James Brentano*
United States Marine Corps — *Brad Triebwasser*
United States Marine Corps — *Mark Johnson*

AUTHOR: **JAMES BRENTANO**, PACIFIC GAS & ELECTRIC

For more information about NAC papers, contact:
Kelli Wiseth, Director, Technical Publications
Network Applications Consortium
c/o Pacific Gas & Electric Company
Mail Code B19A
P. O. Box 770000
San Francisco, CA 94177

Contents

Interoperability: Electronic Mail Systems

Executive Summary

As is the case with many enterprise-wide applications today, e-mail systems employ a variety of competing application programming interfaces (APIs), formats, and protocols. However, in order to work together, or *interoperate*, components must use common APIs, formats, and protocols.

In this paper, NAC analyzes e-mail systems in terms of market segmentation, market penetration, technical completeness, and openness in order to select the most appropriate API implementation from among the competing approaches on the market today. NAC makes recommendations to vendors and consumers in each of eight functional areas that are required for a complete enterprise-wide e-mail system. These areas are:

- Message Exchange
- Client APIs
- Service Management
- Mail Client Management
- Client-to-client
- Directory
- Security
- Time

In the area of message exchange, which by its very nature must occur between different systems, NAC recommends that vendors move toward direct support of SMTP/MIME and X.400 as alternate native transports, and adopt the PEM standard. On the other hand, in the area of client APIs, where there are distinct divisions between operating systems and the vendors that are active in the respective markets, vendors should move to the OS-driven client and server APIs. This means MAPI in the Microsoft Windows environment and AOCE in the Macintosh environment.

Likewise, the client-to-client API area is OS-based; vendors can support OS-based interoperability as a value-added feature now, but should expect it to be a minimum requirement within the next one to two years.

Mail service management is an area where vendors can still jockey for position by participating in the IETF process to refine the proposed standard, but in the meantime provide MIBs for their respective products based on the proposed standard. As far as managing the mail client is concerned, however, vendors should track the desktop management market, particularly DMTF (desktop management task force) in the Windows marketplace, and be prepared to migrate to a standard when the technology matures.

Finally, the areas of directory, security, and time are looking for market leadership and direction, and there is much opportunity for collaboration between NAC and vendors. In the area of directory services in particular, NAC and vendors should work together to define common service APIs and functionality. Likewise, in the area of security services NAC and vendors can work together to develop common requirements and APIs.

In all cases, vendors should begin evolving their existing products to use a common services architecture now, supporting multiple services in the short-run but aggressively adopting market standard when it emerges in the next one to two years.

These recommendations to vendors are aligned with our recommendations to consumers. Thus, recognizing that two alternative APIs may be necessary in the area of message exchange, consumers should identify the applications and locations where SMTP/MIME and X.400 are most appropriate, and begin migrating to one of these as a backbone transport now, as well as move toward the PEM standard.

In the client API area, consumers should adopt OS-based standards — MAPI in the Microsoft Windows desktop and AOCE for the Mac desktop. Likewise, in the area of client-to-client interoperability, use OLE and AOCE as a criteria for purchasing applications when it makes sense to do so.

Consumers should develop expertise in SNMP-based network and systems management, and use the proposed standard mail service MIBs until vendors come out with their implementations of the final standards. In the area of mail client management, consumers should take advantage of existing technology, but don’t expect to leverage that technology in the future as the desktop management standards evolve.

In the area of directory services, consumers can pressure vendors to adopt an interoperable directory standard by making it part of their technology RFP and other purchasing processes. In addition, NAC should work with the key PC e-mail vendors to define common directory service functionality and a set of APIs in developing an X.500 reference model implementation and a set of compliance tests.

In the area of security service, consumers can protect their investments in software by insisting on layered applications that insulate the details of the security service from the base functionality.

Introduction

Nothing sparked the growth of the enterprise network more than the widespread use of e-mail. E-mail has proven to be an effective vehicle for intra- and inter-company communications and a boon to productivity. Unfortunately, along with the proliferation of e-mail came the proliferation of incompatible e-mail products — and the attendant proliferation of APIs, protocols, and formats. The result is that today, the mail frequently doesn't "get through" due to a lack of interoperability among components.

While the solution appears to be adopting standard APIs across all products, the question remains — which ones? As discussed in *Interoperability: A NAC Position Paper*[1], the problem of identifying standards is complicated by the fact that application and service architectures are different; market segments and prevailing market forces are different; and maturity levels, in terms of both technical development and market penetration, are all different among the diverse components that make up an enterprise IT infrastructure.

As a less attractive alternative to standard APIs, a degree of interoperability can be achieved through the use of *gateways*, which translate the protocols and formats used by one application into the protocols and formats used by another. However, gateways degrade performance and should therefore implemented sparingly. They are also limited to use between systems — between two or more different e-mail systems, for example — rather than between the client and server portion of a single application. In addition, gateways demand a great deal of support and maintenance. In fact, in a recent informal survey of NAC members, gateways were identified as the single largest consumer of support resources. For all of these reasons, gateways do not provide a viable interoperability strategy.

To better understand all the issues surrounding e-mail interoperability, we analyzed e-mail systems as follows:

1. We designed a generic e-mail system, summarizing key assumptions about the system, its components, and its basic functionality by defining:
   — an architectural model
   — the underlying structure

2. Based on the generic e-mail system, we defined the components that must interoperate.

---

[1] Published in August, 1994, by the Network Applications Consortium.

3. We extrapolated the interface points by forming a matrix that represents the components as *requesters* or *providers* of e-mail system functions. We also developed a list of the functions that must occur between all components to enable a full-featured e-mail system proposed in the first step above.

4. We compared the interface point matrix with the functional listing and grouped the interface points into functionally similar API sets.

5. We mapped the competing standard APIs, formats, and key industry players to the API sets defined in the step above.

6. Finally, we evaluated existing standard APIs on the basis of market penetration, technical completeness, market segmentation, and openness, and we developed recommendations for vendors and consumers for each API set.

Our focus is on systems that are primarily intended to support the exchange of messages — including text, sound, images, and executable binaries — between human users or their agents. We start with our generic model of an e-mail system, based on the typical functionality available in most products on the market. It should be understood that no general model will accurately represent all of the many products on the market.

E-mail System Architecture

Because e-mail systems can be designed in numerous ways, NAC created a "generic" e-mail architecture to facilitate analysis. Our e-mail architecture implements three key features that any enterprise e-mail system must provide: a client application; a means of storing messages; and a process for moving messages between users across and between networks.

E-mail systems can be architected in a number of ways. For example, the underlying structure of the mail interfaces can be *file-based* or *procedural*. File-based interfaces use the network file system to store messages and address book information, while procedural interfaces follow the client/server programming model. E-mail systems from different vendors have different numbers of components, and even similar functionality may be split up in different ways.

Regardless of the details of implementation, however, an enterprise e-mail system must provide certain basic functions:

- an application which runs on users' desktops and enables them to read, compose, send, and manage messages
- a means of storing messages

- a process for moving messages between users across a network and between networks

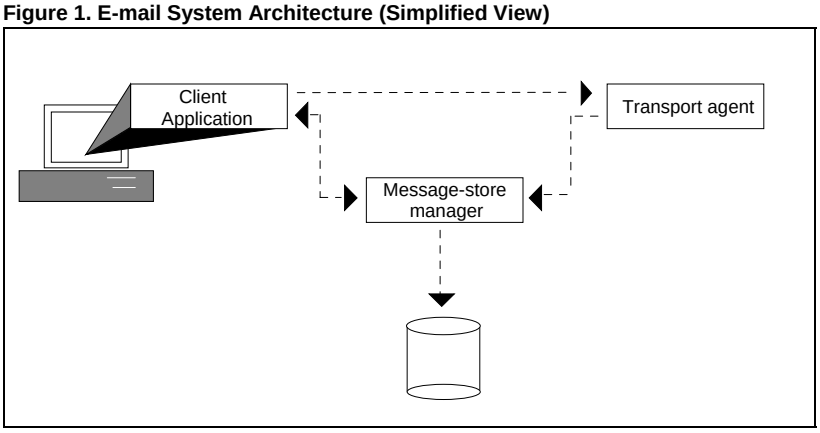
These three basic e-mail functions are shown in the generic model in *Figure 1. E-mail System Architecture* below, and are implemented through the following three generalized components:

- The ***client*** application, which users interact with to access their mailbox, and read, compose, send, and receive messages.

- The ***message-store manager***, which is a software process that controls access to the actual messages, as well as their storage. As shown in the model, the message-store manager is a distinct process, separate from the content of the messages. Although the message-store manager functionality may be implemented in different ways it must be "on the network." That is, a user or process should have access to the message-store manager regardless of location.

- The ***transport agent***, which is a software process that moves the messages between locations. That is, to and from the message-store manager, and to and from other transport agents. Message systems use the "store-and-forward" paradigm in which a message sent from point A to point B may be routed through points C and D. It is the transport agents that perform this routing, by receiving, storing and passing on each message as required.

Figure 1. E-mail System Architecture (Simplified View)



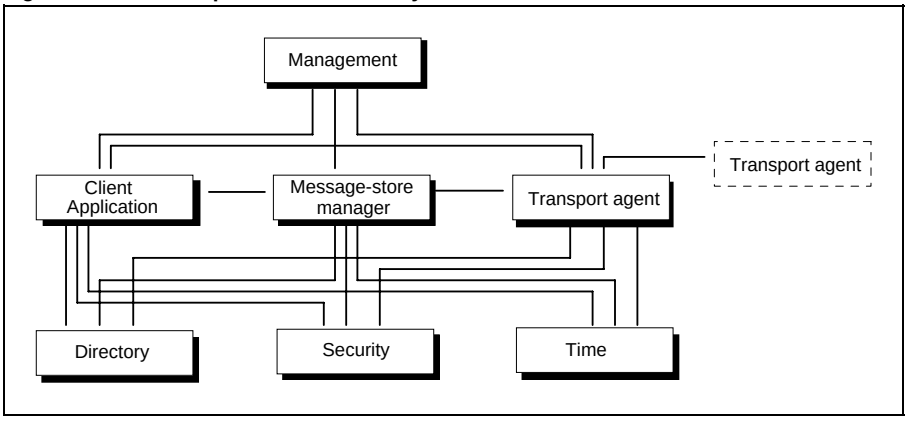
E-mail System Components

Even a very simplified e-mail system requires seven different components that must interact to achieve end-to-end e-mail delivery. Each component comprises a functional unit that must be accessed through an application programming interface.

In addition to the three components described above, a complete e-mail system needs access to several common services, including:

- A ***directory service***, which is an electronic address book that enables users to select e-mail recipients and address their messages. The directory service is also required to resolve user-friendly names, such as "John Brown," into e-mail addresses such as jbrown@sales.ca.greed.com.
- A ***security service***, which ensures e-mail privacy by *authenticating*, or verifying, that the user attempting to access John Brown's mail is in reality John Brown (or at least knows John Brown's password). Note that message security — that is, encryption — is provided by the client since it must be end-to-end.
- A ***time service***, which enables the mail service to time-stamp messages, an important feature in terms of auditing and management.
- A ***management system*** that supports administration and management of users, mailboxes and mail services. This is particularly important in an enterprise environment where there may be thousands of mail users and dozens, if not hundreds, of mail servers.

Figure 2. Seven Components of E-mail System



Thus, a total of seven components, potentially provided by seven vendors, must interoperate. Each component comprises an individual functional unit that is accessed by the other components by means of a standard interface. If each component were to interact with every other component — including different instances of the same component — forty-nine standardized interfaces would be required.

However, not all forty-nine of these interfaces are required or fall within the scope of this paper. Some components never need to call each other, or the relationship is uni-directional; for example, the e-mail client never calls for services from the management system, so an interface from the management system to the mail client isn't required.

And although some interfaces — such as the interface between the directory service and the security service — might generally be required, they don't involve any of the three e-mail system components and thus are beyond the scope of a paper on e-mail interoperability. Thus, only nineteen of the forty-nine possible interface points need further analysis.

E-mail Interface Matrix

Describing the interface points as a matrix of rows and columns enables a clearer view of the nineteen distinct APIs — sets of function calls that a server application provides to client applications — that are necessary for an interoperable enterprise e-mail system. A listing of the generic functions that must provided is in the Appendix.

The matrix below displays the forty-nine possible interface points and identifies the nineteen with which we are concerned. Each of these interfaces is implemented as an API through either static or dynamic libraries. An API is a set of function calls that a service *provider* publishes, or exports, and which client applications (*requesters)* use to gain access to the provider's services. Thus, the columns in the matrix below denote the requester and the rows denote the provider.

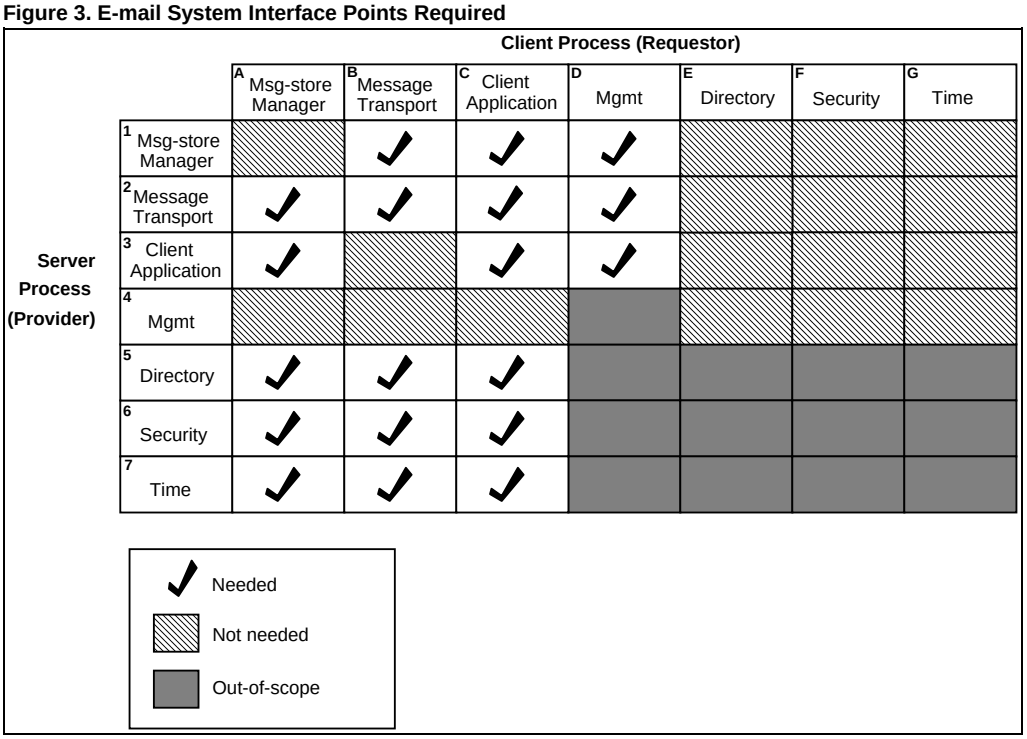
The basic APIs needed to provide a functional e-mail system are listed below. The alphanumeric identification in parentheses refers to the intersection point on the matrix shown in Figure 3 below.

- 1. Message-store manager to message transport (A2)
- 2. Message-store manager to client application (A3)
- 3. Message-store manager to directory (A5)
- 4. Message-store manger to security (A6)
- 5. Message-store manager to time (A7)
- 6. Message transport to message-store manager (B1)
- 7. Message transport to message transport (B2)
- 8. Message transport to directory (B5)
- 9. Message transport to security (B6)
- 10. Message transport to time (B7)
- 11. Client application to message-store manager (C1)
- 12. Client application to message transport (C2)
- 13. Client application to client application (C3)
- 14. Client application to directory (C5)
- 15. Client application to security (C6)

- 16. Client application to time (C7)
- 17. Management-system to message-store manager (D1)
- 18. Management to message transport (D2)
- 19. Management to client application (D3)

As an example of how to read the matrix: The client (requester) will need access to the message-store manager, so the message-store manager (provider) must provide an appropriate interface (C1). However, the message-store manager (requester) will have no need to access the management system (provider), so no interface is needed and the box at that juncture (A4) is grayed-out.

Figure 3. E-mail System Interface Points Required



In conjunction with developing this matrix, we developed a generic listing of the functions that must be provided at each interface (see *Appendix A. Generic Functionality of E-mail Interface Points*). The functions are used to evaluate the completeness of individual APIs and proposed standards; the listing can also be used by consumers to evaluate a product's functionality when making a purchase.

Functional Analysis

An analysis of the interface points and the functional listing reveals a logical grouping of eight distinct sets of functions that enable end-to-end enterprise e-mail

Each of the interfaces required for a complete end-to-end e-mail system, shown in Figure 3 above, is embodied in a set of APIs. Unfortunately, not only do vendors use syntactically different APIs, they also use different application models.[2] Thus, even the functionality provided by the APIs is different.

To get around this problem, we identified the generic set of functions required at each interface , in the matrix (see Figure 3 above and *Appendix A. Generic Functionality of E-mail Interface Points*). This analysis reveals many opportunities for grouping the APIs together, thereby reducing the number of standards required.

For example, the directory functions used by the message-store are very similar to those that the mail client uses. Thus, in this case, it is useful to think in terms of a single directory service API which exposes the full functionality of the directory service.

Applying the same approach to the entire matrix yields eight API sets which need to be standardized to guarantee interoperability in the e-mail arena. These functional areas are described as follows:

- **Message Exchange** includes both the transfer protocol and the format of the message, and provides independence between the sender and the receiver of a message. (A2, B1, B2)
- **Client APIs** which allow the desktop client application to be developed and used without being tied to a specific mail service provider. (A3, C1, C2)
- **Mail Service Management API** which allows a management tool to be chosen independent of the mail service which it is managing. (D1, D2)
- **Mail Client Management API** which allows the mail client to be managed by any desktop management application. (D3)
- **Client-to-client Application Interoperability APIs** which allow a desktop application to invoke and communicate with the mail client regardless of the vendor of either application. For example, a Lotus spreadsheet invoking a Microsoft Mail front end. (C3)

---

[2] See *E-mail Architecture* earlier in this document.

- **Directory Service APIs** which allow the mail client or service to take advantage of any directory service. (A5, B5, C5)

- **Security Service APIs** which insulate the mail applications from the implementation of the authentication services. (A6, B6, C6)

- **Time Service API** which provide a standard way for mail applications to get reliable time stamp regardless of the provider. (A7, B7, C7)

Competing Implementations of the API Sets

The competing standard APIs, formats, and key industry players are mapped to the API sets defined above.

The list below displays some of the proprietary and standard interfaces currently implemented in products. As the list makes clear, the current market consists of many incompatible, much less interoperable standards. The standards listed also highlight the segmentation of the market. This segmentation, for example between the Windows, Unix and Macintosh markets, is critical since, as was discussed in *Interoperability: A NAC Position Paper* most effective standards emerge from the marketplace.

Figure 4. Functional API Sets

	Message Exchange	Client APIs	Mail Service Management	Mail Client Management	Client-to-client Application	Directory	Security	Time
Apple	AOCE	AOCE			AOCE, OpenDoc (CIL)	AOCE (PowerShare Catalog)		AOCE (PowerShare)
Banyan	VINES API	VINES API	Mnet/Mmail			StreetTalk (STDA)	VINES Security	VINES time
Lotus	cc:Mail Router; Notes LCS	VIM Notes	cc:Mail Notes			cc:Mail, Notes address books	cc:Mail, Notes security	
Internet	SMTP/MIME	POP, MH	Sendmail		Unix IPX, XWindows	DNS MX	etc/hosts, etc/passwd,	NTP
Microsoft	Mail	MAPI	Mail		OLE1, OLE2, DDE	Mail address book	Mail security	DOS/Windows time; NT time
Novell/ WordPerfect	MHS Router, GroupWise router, OME (open message environment)	SMF71, AOCE, MAPI	NetWare Mgmt, GroupWise	GroupWise	OpenDoc (CIL), OLE1, OLE2, DDE, AOCE	Global MHS Directory, NetWare NDS, GroupWise directory	GroupWise security	NetWare time, GroupWise time
OSF						CDS	DCE Security	DCE Time service
X.400	P1, P2, P3 & IEEE 1224	P7 & IEEE 1224	CMIP			X.500	X.400 security svcs, X.509	
Other	SoftSwitch	CMC		DMI			Kerberos	

Evaluation of Standard API Sets

A brief discussion of the market and technical criteria for evaluating existing standard APIs. NAC recommends different strategies, for vendors and for consumers, in each of the eight respective API sets.

As the table above graphically demonstrates, vendors are a long way from providing interoperability. Interoperability depends on a single API at each interface, so that any client (requester) can operate with any server (provider). As is pointed out in *Interoperability: A NAC Position Paper*, it is primarily the marketplace which chooses an emerging API as a "winner" from a market of competing standards. The goal is to identify early an API which all vendors will adopt, since this is what provides interoperability. Thus, NAC evaluates each of the various proposed standards on the basis of the following four criteria:

- market segmentation
- market penetration
- technical completeness
- openness

Before evaluating on the basis of the other three criteria, the field of competing APIs must be narrowed in terms of appropriate ***market segmentation***, given that the "IT industry" is in fact composed of several smaller markets.[3] Moreover, most vendors operate in only one or two of these market segments. Some of the more obvious segments are the Unix/Open Systems, the PC, the SQL RDBMS, and the corporate applications markets, for example.

Although it's reasonable to hope for the emergence of common, standard APIs, within each of these market segments, it's uncommon for standards to cross market-segment boundaries. For example, although the PC platform market — which includes the Apple Macintosh — is the most critical market segment both because of its size and of its continuously improving price/performance ratio, it's highly unlikely that an API selected as the winner in the PC market will be adopted in the Unix market.

After segmenting the market into its constituent submarkets, the viable API sets can be described in terms of their ***market penetration***. Market penetration is the

---

[3] See *The NAC Marketplace Model for Selecting a Standard* in *Interoperability: A NAC Position Paper* for further discussion of marketplace issues.

measure of the acceptance by the market. Because of the "lock-in" phenomenon[4] in the market, the standard that has early market dominance is likely to prevail.

Market penetration alone is not enough, however. The standard API set must also be ***technically complete***. This isn't to say that the eventually dominant standard will be the *best* technical solution, but rather that it must provide enough functionality so that ISVs (independent software vendors) will gain, rather than lose, by adopting the standard. That is, the marginal benefit to ISVs of using the standard — and thereby extending their market reach — is greater than the marginal cost of using a less-than-perfect API set.

Finally, ***openness*** is another important factor: vendors are much more likely to embrace a standard that is not owned by one of their competitors. At least three levels of openness can be identified[5]:

- public ownership and published specification
- private ownership, but published specification
- private ownership and unpublished specification

In some respects, public ownership is preferred. However, a completely public process also has drawbacks, as demonstrated by the failure of the OSI standards. Nonetheless, for an API to be considered as a potential industry standard, the specification must at minimum be published, whether the standard is owned by the public or is private. It must also be generally and internationally available with reasonable licensing fees.

* * *

---

[4] This "lock-in" phenomenon is the same one which, for example, accounts for the continued use of the "QWERTY" keyboard.

[5] Although NAC's approach to the open standards discussion is a departure from the approach presented by the Burton Group, with its open-closed-public-proprietary model, we don't dispute the Burton Group model. However, the Burton Group tool enables an IT manager to assign vendors to different areas of the open-closed grid based on their "openness," while NAC's approach is to select distinct APIs with a view toward migrating them to the public arena independently of a vendor's position relative to the question of openness.

In the discussions that follow, we address each of the eight distinct interface areas. Such separation is necessary because, as just described, each area is potentially at a different level of both technical and market maturity, and because each may exhibit different market dynamics.

Message Exchange

Message exchange APIs support the transfer of e-mail messages from one system to another; that is, from the message-store manager to a transport agent or from one transport agent to another. Message exchange APIs specify how one process makes the connection with a peer process, and how the processes send and receive messages, respectively. To guarantee that the receiving process will understand the message, message exchange APIs also prescribe the format of the message. And to ensure end-to-end message security and integrity, the message exchange APIs include provisions for message encryption. In an e-mail setting, encryption serves several purposes: privacy, integrity; authentication; and non-repudiation.

A combination of the two alternative encryption methods — public key and private key — is generally acknowledged as providing the optimal trade-off between security and efficiency. Public key encryption requires that the sender and receiver of an encrypted mail message each have access to the other's (authenticated) public key. While key management should be handled by the directory service, interoperable message encryption requires the use of both a standard encryption algorithm and a common key management methodology by both the sender and the receiver.

By its very definition, message exchange is expected to occur between environments, and hence, between markets. For example, mail sent from a PC-LAN environment must be able to be received, de-crypted, and displayed by a process or user running in a Unix multi-user environment.

This cross-market nature of message exchange prevents any single vendor from dominating the market. In fact, most of the vendors competing in this area provide gateway solutions rather than supporting an end-to-end standard. Perhaps the one exception is Lotus' cc:Mail, which provides access to most environments but through a private and unpublished API.

Because the requirements and functionality in this area are well defined and understood, and because no single vendor solution dominates, de jure standards, promulgated by standards bodies, are more appropriate than de facto vendor standards. The two candidates for base message exchange standards are the X.400 specification from CCITT and the SMTP/MIME (simple mail transport protocol/multipurpose internet mail extension) specification from the IETF

(Internet engineering task force). Both of these standards are currently in use and each is important in different market segments.

X.400 is the dominant standard outside the United States; it is supported by the commercial e-mail interchange providers, such as the inter-exchange telephone companies. In addition, X.400 has a more sophisticated Electronic Document Interchange (EDI) standard (X.435) than the SMTP suite, making it most appropriate for wide use in electronic commerce than SMTP, even in the United States.

Nonetheless, the SMTP protocols have been delivering mail successfully for nearly two decades, and although MIME is a relatively recent addition to the protocol suite, SMTP is used in millions of sites. With its close ties to the ARPA TCP/IP protocol stack — which has emerged as *the* de facto networking standard — SMTP is already the de facto mail transport standard on the Internet, which is the foundation for the National Information Infrastructure (NII) and the Worldwide Information Infrastructure (WII) also known as the "information superhighway."

Furthermore, most PC-LAN-based e-mail systems already support SMTP, (although not yet MIME). And because e-mail access to the Internet through SMTP is relatively inexpensive, SMTP usage can be expected to continue to grow rapidly in both the home and small business markets.

Encryption standards are less mature than those for mail interchange, although it is generally accepted that e-mail encryption should be based on a public key model. The Pretty Good Privacy (PGP) standard is available on many platforms, but its legal history has caused confusion: although the current version is not believed to be in violation of any patent protections, prior versions may have been. The Privacy Enhanced Mail (PEM) standard is the emerging Internet standard. RIPEM (Riordan's Internet Privacy Enhanced Mail) is an implementation of PEM which is fairly widely available. In the PC market, Lotus Notes provides public key encryption, but in a closed architecture.

For the same reasons that SMTP/MIME continues to grow in acceptance — cross-platform functionality at a reasonable price — PEM will likely emerge as the standard for mail security. However, before any encryption standard can be unilaterally endorsed, the United States export restrictions on some encryption technologies need to be modified. While it is possible to provide secure international e-mail despite these restrictions, they do make the process more difficult.

Recommendations to NAC members and other consumers:

- Identify the applications and locations where SMTP/MIME and X.400, respectively, are most appropriate.
- Begin migrating to SMTP/MIME and X.400 as backbone transports now.
- Begin migrating to the PEM standard.
- Support legislation that will eliminate or ameliorate U.S. export restrictions on encryption technology.

Recommendations to vendors:

- Directly support SMTP/MIME and X.400 as alternate native transport interfaces.
- Adopt the PEM standard.
- Phase-out existing proprietary transport and security mechanisms.
- Lobby for legislation that will eliminate or ameliorate U.S. export restrictions on encryption technology.

Client APIs

Client APIs are used by developers to create desktop applications that take advantage of the services provided by the message-store manager (and indirectly by the message transport). In addition to functions such as submitting and reading messages the client APIs define the format of the message-store and of messages as seen by the client application.

The client APIs have recently been the subject of the widely discussed "API wars.[6]" What is not always understood is that these API wars represented only one market segment: PCs running Microsoft Windows. Other markets including the DOS, Unix, and Macintosh markets were not directly involved. This narrowness of focus points out the significance of market segmentation in this

[6] Confrontation between Microsoft, which supports MAPI, and Lotus, which supports VIM. See the Burton Group, *Electronic Messaging* discussion under the *Network System Services* track for an excellent discussion of the API wars and related market issues.

area. Some cross-platform client API standards have been proposed, such as the Common Mail Calls (CMC), the Post Office Protocol (POP), and the P7 Message Store Protocol. However, CMC and POP are minimal subset APIs that lack much of the functionality required for a full-featured mail client, particularly in their ability to manipulate messages in the message-store (although the CMC APIs may be appropriate for the class of "mail enabled" applications which require only limited services from the mail store). And P7 has yet to be adopted in the PC market, even by those vendors who support X.400 based transport between MTAs.

The reality of the market is that mail client application development is still very much operating system (OS)-dependent, not due to technical issues of mail itself but rather because of the numerous other dependencies that any desktop application has on the underlying operating system and presentation environment. Given this reality, it doesn't make sense to talk about cross-platform client API specifications because it isn't realistic to develop a cross-platform client. This means that service providers will have to provide interfaces for each of the major client markets.

Another reality is that the OS vendors control the desktop market. Given this control of the market as well as the close ties between the client applications and the operating system services, it is inevitable that the client APIs endorsed by the OS vendors will emerge as dominant. In the Microsoft Windows environment this means MAPI, in the Macintosh environment AOCE will dominate, and in the Unix environment Sendmail is the standard.

Recommendations to NAC members and other consumers:

- Adopt OS-based standards for mail client APIs.

Recommendations to vendors:

- Move to OS-driven client and server APIs — MAPI in the Microsoft Windows environment and AOCE in the Macintosh environment.
- If you own the APIs, release them into the market to hasten the inevitable.
- If you don't own the APIs, wrest control of them away from the owners.

Service Management

Service management APIs provide functionality such as starting and stopping the service, managing message queues, and monitoring the performance of the messaging subsystem.

System management continues to be one of the hot issues in the information technology industry. It is clear that effective integrated management of mail services (message-store and transport) is a prerequisite for the successful, cost-effective implementation of interoperable mail systems. However, mail is just one of the many services that need to be managed in a heterogeneous environment. Thus it is the network and systems management market that is the critical market segment.

The management market is controlled by network management vendors (Sun, HP, IBM, Tivoli, for example). The simple network management protocol (SNMP) has emerged as the defacto standard in this market; more specifically, SNMPv.2 management via management information bases, or MIBs. The IETF has a proposed standard MIB for Mail Transfer Agents (RFC 1566). This MIB definition will evolve through the IETF standards process model[7] with the participation of the major mail system vendors (Novell, Banyan, Lotus, and Microsoft).

Recommendations to NAC members and other consumers:

- Develop expertise in SNMP-based network and systems management.
- Use the proposed standard mail service MIBs until vendors come out with their implementations of the final standards.

Recommendations to vendors:

- Provide MIBs based on the proposed standard for your products immediately. Participate in the IETF process to refine the proposed standard (RFC 1566).

Mail-client Management

Mail client management APIs are expected to provide management software with information about the status of the mail client including whether it is executing, and if so, from where, what version is running, and so forth.

As with service management, mail client management is one aspect of a larger management issue, in this case desktop management. However, unlike the service

---

[7]See *Interoperability: A NAC Position Paper* for a full discussion of various standards-adoption processes.

management market, the desktop management market is very immature. Products are emerging in each of the operating system based market segments. And the Desktop Management Task Force's (DMTF) Desktop Management Interface (DMI) has some promise as a consensus-based standard. A cross-platform standard will be important since management is by its nature deals with heterogeneous environments.

Management of the mail client can be expected to follow the rest of the desktop software. However the market is still too immature to speculate on the eventual outcome.

Recommendations to NAC members and other consumers:

- Take advantage of any existing mail client management, but expect it to be a stranded investment.

Recommendations to vendors:

- Track the desktop management market.

- Provide proprietary client management as demanded by the market, but with an eye to migrating to a standard when the technology matures.

Client-to-Client

The client-to-client APIs are those provided by the mail client to allow other applications running in the same desktop to take advantage of mail functionality by invoking the mail client, either on screen or in the background. This is particularly important in the PC environment where integrated desktop applications are the norm, and where users will expect to see the same interface whether they invoke a mail client from their spreadsheet or from their word processor.

Like the mail server to mail client APIs, this market is very much segmented by operating system. The inter-application connectivity will be provided by the OS vendors: DDE and OLE2 by Microsoft, AOCE by Apple, and Spec 1170 IPC by the Unix vendors. While these protocols define how applications find each other, and how they communicate, they do not define the content of the communication. For example, OLE2 does not define what functionality a mail client exports to a spreadsheet.

This is particularly significant with the recent emergence of the "office suite" as the dominant marketing method for PC desktop software. If this trend in marketing continues, we can expect to see each of the major PC office vendors

(Lotus, Microsoft, and WordPerfect) provide their own “mail enabling” APIs which their other desktop products take advantage of. Smaller mail client developers will be forced to develop different application versions for each of the three major office suites.

Recommendations to NAC members and other consumers:

- Use client-to-client interoperability based on OLE and AOCE as a criteria for purchasing applications when it makes sense to do so.

Recommendations to vendors:

- Support OS-based application interoperability as a value added feature immediately, but expect it to become a minimum requirement in the next one to two years.

Directory

The directory service APIs provide name lookup and name resolution; map user names to the associated mail service and network address; and acts as a repository for public encryption keys.

Many of the mail products on the market use proprietary special purpose directories (often called address books) to provide name lookup services. However, this is not a sustainable method as more and more applications need access to directory services.

While there are a number of “open” standards (for example, DCE/CDS, X.500, DNS) in the market, acceptance has been slow. The exception is DNS which is widely installed, but does not provide user name lookup services. What’s perhaps more important, none of the open directory standards has been successfully ported to the PC platform by a major vendor. This may be in part because all of these standards come from the Unix environment and neither the technology nor the marketing are designed for the PC network market. The same is true of the proprietary Unix directory services such as Sun’s NIS+.

On the other hand, the PC-LAN-based directory services such as the Novell Bindery and the NTAS Domain have not, in general been designed from an enterprise perspective and don’t scale well in multi-server, multi-company environments.

Lotus Notes provides a directory service which is well integrated with both the database and mail functionality, which manages public keys, and which scales well, but it has very little market share and is a closed system. Banyan’s

StreetTalk has for several years been the most technologically advanced directory service in the PC network arena but Banyan has so far failed to capitalize on this lead. In the meantime Novell has recently entered the market with NDS. Although NDS has been slow getting started, Novell's huge LAN market share makes NDS a potentially important product.

Each of the four key PC e-mail vendors — Microsoft, Lotus/SoftSwitch, WordPerfect/Novell, and Banyan/Beyond — have products (or plans) for X.500-"like" directories. However, what will still be needed is an interoperable, realistic, implementable PC-market-based API standard interface to these pseudo X.500 directories. Although it is still too early to tell, the Lightweight Directory Access Protocol (LDAP) may emerge as the standard way of accessing a variety of proprietary directory implementations. In addition to agreement on the nominal standards, one of the things which is needed is a certification test suite to allow verification of compliance and interoperability.

Recommendations to NAC members and other consumers:

- Individual companies should pressure vendors to adopt an interoperable directory standard by making it part of technology RFPs and other purchasing processes.

- NAC should work with the key PC e-mail vendors to define common directory service functionality and a set of APIs in developing a directory interface reference implementation and a set of compliance tests.

Recommendations to vendors:

- Work with NAC to define common directory service APIs and functionality.

- Begin evolving existing products to use a common directory.

- Develop new products using a common services architecture.

Security

The principle function of the security service APIs is to provide an authentication mechanism. The security service and the directory service need to be tightly linked. The market issues in this area are very similar to those surrounding the directory service. The solutions from the Unix world have not migrated to the PC world, because of both technical and market incompatibilities. Open standards such as MIT's Kerberos and the DCE Security service have "mind share" but are still not deployed in the high end distributed computing environment, let alone in the much more resource constrained environment of PC networks. Lotus Notes

provides a successful security implementation including secure and signed mail, but it is a closed system and Lotus has not provided a migration path. Banyan's security service is tightly integrated with both its directory service and its mail service, and it has been demonstrated to scale. However, as with the directory service, Banyan has not effectively marketed its security service..

In the meantime RSA and others are providing pieces of the security solution but it is still basically a "roll-your-own" environment in which interoperability and security-service-based authentication are fundamentally incompatible.

Recommendations to NAC members and other consumers:

- Protect large software investments by insisting on layered applications which insulate the details of the security service from the base functionality.
- Expect trade-offs between security and interoperability for the next few years.
- Address the security issues with a NAC SIG and provide vendors and other consumers with a focused technical recommendation along the lines of that proposed for directory services above.

Recommendations to vendors:

- Work with NAC to develop a common set of requirements and APIs for a scalable security service.
- Begin evolving existing products to a common services architecture now.
- Develop new products using a common services architecture.
- Support multiple services and supply your own services in the short run.
- Aggressively adopt a market standard when it emerges in the next one to two years.

Time

The time service APIs provide an authoritative time source to applications. Time stamps are used throughout electronic mail systems and are especially important in electronic commerce applications, and in auditing.

In some environments the presence of a time service is hidden from the application developers by the OS. The developer simply uses the native OS time calls, while the OS itself gets its time from the time service.

There are a number of time service standards in each market. However, the emergence of a dominant standard is likely to follow the outcome of the directory and security standards.

Recommendations to NAC members and other consumers:

- Address time service issues on a case by case basis.

- Insist on well-layered applications that will be able to take advantage of a standard time service when it emerges.

Recommendations to vendors:

- Build applications which assume the existence of a time service, even if it's necessary to build a dummy time service as well.

Conclusion

In order to better understand the issues surrounding e-mail interoperability, NAC analyzed several leading e-mail systems on the market today. Our analysis confirms the importance of a common-services-based architecture and highlights the fact that the cost and unreliability of multiple, special purpose — and proprietary — services is too high. It is NAC's position that interoperability is a critical concern in this environment, and we encourage vendors to work toward the specific recommendations made in this document. For example, vendors should adopt MAPI as the client interface standard in the Windows environment and AOCE as the standard in the Macintosh environment. In addition to making recommendations to vendors and consumers, we have identified areas where NAC and vendors can profitably work together, for example, developing a reference implementation and interoperability standards for an X.500 like e-mail directory.

NAC recognizes that there is no "silver bullet" that will resolve the issues of e-mail interoperability. Market segmentation is a key issue, and the Windows PC market is the dominant one. Potential solutions that exist in the Unix/Open Systems environment must overcome significant technical and marketing issues if they are to be generally effective in this environment. This seems to be occurring in the relatively open area of message exchange, but not yet in the area of common services. It is unlikely to happen at all in the areas that are tied closely to the desktop operating system, such as the mail client APIs and inter-

application interoperability. In general, both vendors and consumers must recognize that the market — not the technology — is the driver, and the key to success is understanding and anticipating the market, rather than resisting it. Consumers who are successful at identifying and implementing standards early will reduce the risk of stranding their current and future IT investments.

Likewise, vendors who adopt emerging standards early will gain sales, particularly in the corporate market where buyers are increasingly intolerant of closed and proprietary implementations that lock them in to a single vendor. Market share in the maturing e-mail market will to a large extent be based on openness and support for interoperability. In this paper and in future work with vendors, NAC hopes to facilitate the development of interoperable products in the network arena. We welcome your comments and suggestions about the content of this paper. Please address your remarks to:

James Brentano, Senior Technology Planner
Computer & Telecommunications Services
Pacific Gas & Electric Company
Mail Code V22C
P.O. Box 770000
San Francisco, CA 94177
jabu@pge.com

References

Alt.Security.PGP Frequently Asked Questions. Version 9. Gary Edstrom 1994/4/17

Apple Computer Inc. et al. *Vendor Independent Messaging Interface Functional Specification Version 1.00*. 1992.

Banyan Systems Incorporated. *VINES Mail Client Programming Interface*.

Black, Uyless. *The X Series Recommendations*. McGraw Hill, New York. 1991.

Borenstein, N., and Freed, N. *MIME (Multipurpose Internet Mail Extensions) Request for Comments*: 1521 Network Working Group. September 1993

Burton, Craig and Lewis, Jamie. *Messaging Interfaces: MAPI, VIM, XAPIA/CMC, NetWare SMF71*. Burton Group Report. October 1993.

Client Developer's Guide: Messaging Application Program Interface Software Development Kit, Version 1.09. Microsoft Corporation. 1993

Costales, Bryan, et al. *Sendmail*. O'Reilly and Associates, Inc. Sebastopol, California, 1993.

Developer's Guide: Common Messaging Call Application Programming Interface Version 1.09. Microsoft Corporation. 1993.

Directories, Naming and Enterprise Electronic Mail. Soft-Switch Inc., March 21, 1994.

Goodwin, Tim. *MIME Frequently Asked Questions (FAQ) Version: 3.4*. Last-modified: 1994/02/10

Grand, Mark. *MIME Overview*. Revised October 26, 1993

Kerberos Users' Frequently Asked Questions. Compiled by: Barry Jaspan. August 25, 1993

Linn, J. *Privacy Enhancement for Internet Electronic Mail: Request for Comments: 1421*. Network Working Group. February 1993

Mail Monitoring MIB Request for Comments: 1566 Kille, S. and Freed, N. Network Working Group, ISODE Consortium. January 1994

Malamud, Carl. *Analyzing Novell Networks*. Van Nostrand Reinhold, New York, 1990.

Mercilliot, Marc, Carr, Eric, and Anderson, Ron. *Why is E-Mail Still So Bad?* Network Computing. May 1 1994.

Morse, Stephen. *Microsoft Mail Directory Synchronization*. Network Computing. January 15, 1994.

NetWare Global MHS and NetWare 4.0. Novell Inc. 1993.

Postel, Jonathan B. *Simple Mail Transfer Protocol. Request for Comments* 821. August 1982

Provider Developer's Guide: Messaging Application Program Interface Software Development Kit, Version 109. Microsoft Corporation. 1993.

Radicatti, Sara. *Electronic Mail: An Introduction to the X.400 Message Handling Standards*. McGraw Hill, New York. 1992.

Radicatti, Sara. *X.500 Directory Services*. Van Nostrand Reinhold, New York. 1994.

Rose, M. *Post Office Protocol - Version 3, Request for Comments 1225*. Network Working Group. May 1991

Rose, Marshall T. *The Little Black Book: Mail Bonding with OSI Directory Services*. Prentice Hall, Englewood Cliffs, NJ. 1992.

Rosenberry, Ward; and Teague, Jim. *Distributing Applications across DCE and Windows NT*. O'Reilly and Associates, Inc. Sebastopol, California, 1993.

Rosenberry, Ward; Kenney, David; and Fisher, Gerry. *Understanding DCE*. O'Reilly and Associates, Inc. Sebastopol, California, 1992.

Spec 1170 Explained and Analyzed. Uniforum Monthly. April 1994

Standard for the Format of ARPA Internet Text Messages. Request for Comments 822. Revised by Crocker, David H. August 13, 1982.

Van Heyningen, Marc. *RIPEM Frequently Asked Questions*. July 31, 1993

X.400 API Association. *Common Messaging Call API*. Version 1.0. June 1, 1993. Mountain View, California.

Yeong, W., Howes, T., and Kille, S. *X.500 Lightweight Directory Access Protocol: Request for Comments 1487*. Network Working Group. July 1993

Appendix A. Generic Functionality of E-mail Interface Points

Disclaimer: This functionality is simply a representative sample used for the sake of analyzing the functionality common to a number of mail APIs and is not represented as being either complete or consistent.

FUNCTIONALITY EXPORTED BY MESSAGE-STORE MANAGER

USED BY MESSAGE TRANSPORT

Deliver

submit envelope
submit header
submit message
submit attachment
submit report
do you accept messages from this id
do you accept messages to this id

USED BY MANAGEMENT SOFTWARE

Session

Mailbox

create mailbox
delete mailbox
count mailboxes
enumerate mailboxes
rename mailbox
empty mailbox
move mailbox

Messages

count messages
get message header
delete message
set message delivery priority

User

get disk space used
set disk usage quota
get user stats

Misc.

start service
stop service

USED BY CLIENT APPLICATION

Read messages

set search criteria
get search criteria
set sort criteria
get sort criteria
get message count
enumerate message headers
get message body part count
get message body part size
get message body part
get attachment count
get attachment size
get attachment
get message flags
mark message as read/unread
move message
delete message

Create message

create new header
add/delete recipient
set message flags
set message header text
set message body part
delete message body part
set attachment
delete attachment
send message
send with deferred delivery
save message
submit probe

Forward

initialize forward message
set forward recipients
set forward comments

Folders

create folder
delete folder
set receive folder
get receive folder
rename folder
move folder
list folders

Mailbox

empty mailbox
get mailbox stats
set defaults
get defaults

Misc.

get user preferences
set user preferences
register for new message notification
unregister for new message notification
cancel deferred delivery

FUNCTIONALITY EXPORTED BY MESSAGE TRANSPORT

USED BY MESSAGE-STORE MANAGER AND MAIL CLIENT

Handshaking

get address types supported
get data types supported

Message submission

submit envelope
begin message
end message
begin attachment
end attachment

USED BY OTHER MESSAGE TRANSPORTS

Handshaking

get address types supported
get data types supported

Message submission

submit envelope
begin message
end message
begin attachment
end attachment

USED BY MANAGEMENT SOFTWARE

Queue management

list queue contents
delete queued message
get queue and service statistics

Message management

get message envelope
get message header
change message priority

Misc.

get log data
start service
stop service

FUNCTIONALITY EXPORTED BY CLIENT

Used by Message-store Manager
notify

Used by Management software
ping
report version
report status

Used by other Client applications
create message
set message recipient
set message header
set message body
set message attachment
send message

FUNCTIONALITY EXPORTED BY DIRECTORY

USED BY MESSAGE-STORE MANAGER

Client/server rendezvous
register this service (set port)
find other services (get port by service name)

Name resolution
resolve name (alias)
resolve name list
get recipient's mail address

USED BY MESSAGE TRANSPORT

Client/server rendezvous
register this service (set port)
find other services (get port by service name)

USED BY CLIENT APPLICATION

Session
open
close

Display names
get first name (1st in directory)
get next name
get last name (last in directory)
set search/display criteria
get search/display criteria

Attributes

get attribute count
list attributes
get attribute value

Name resolution

resolve name (alias)
resolve name list
get recipient's mail address

Client/server rendezvous

register this service (set port)
find other services (get port by service name)

FUNCTIONALITY EXPORTED BY SECURITY SERVICE

USED BY MESSAGE-STORE MANAGER

Authentication

login
logout
authenticate requester

USED BY MESSAGE TRANSPORT

Authentication

login
logout
authenticate requester

USED BY CLIENT APPLICATION

Authentication

login
logout
authenticate requester

Message security

get public key
set public key

FUNCTIONALITY EXPORTED BY TIME SERVICE

USED BY MESSAGE-STORE MANAGER, MESSAGE TRANSPORT, AND CLIENT

get local time
get GMT

Appendix B. Acronym Glossary

ANSI	American National Standards Institute. A leading United States standards-setting organization, and a member of CCITT.
AOCE	Apple Open Collaborative Environment.
API	Application Programming Interface. An API is the formally defined programming language interface to a service or application's functions.
ASCII	American Standard Code for Information Interchange. A system for representing alphanumeric data using seven-bit data string; one of two such systems used in data transmission between computers.
CCITT	Consultive Committee for International Telephony and Telegraphy. The international body that sets telecommunication standards.
CMC	Common Messaging Calls. A standard promulgated by the XAPIA.
DCE	Distributed Computing Environment.
DDE	Dynamic Data Exchange. A feature of Microsoft Windows operating environment and supported applications.
DME	Distributed Management Environment.
DMI	Desktop Management Interface.
DMTF	Desktop Management Task Force.
DOS	Disk Operating System.
EBCDIC	Extended Binary Coded Decimal Interchange Code. An 8-bit data-exchange code used in computer systems and associated communications equipment.
EDI	Electronic Data Interchange.
FDDI	Fiber Distributed Data Interface. A LAN technology that permits 100-megabits-per-second (Mbps) data transfer; proposed as ANSI standard X3T9.5 and based on fiber optic cable.
FTP	File Transfer Protocol. An upper-level TCP/IP service that allows copying of files across the network.
HDLC	High-Level Data Link Control. The ISO's physical-link protocol. Various manufactures have proprietary versions: an example is IBM's synchronous data-link control (SDLC).
HLLAPI	High Level Language Application Program Interface. HLLAPI is a programming interface that allows an application program running on a PC to communicate with an application running on an IBM mainframe.

ICMP	Internet Control Message Protocol. The TCP/IP process that provides the set of functions used for network layer management and control.
IDAPI	Integrated Database Application Programming Interface. A database API, similar in concept to Microsoft's ODBC, but from competing vendors such as Borland and Novell.
IEEE	Institute of Electrical and Electronics Engineers. A professional organization that sets international networking standards.
IETF	Internet Engineering Task Force.
IP	Internet Protocol. The routing part of TCP/IP. An IP datagram is the basic unit of information passed across the Internet.
IPX	Internetwork Packet eXchange. A transport protocol found in Novell NetWare networks.
ISDN	Integrated Services Data Network. ISDN is an international telecommunications standard that allows a communications channel to simultaneously carry voice, video and data.
ISO	International Standards Organization. An independent international body formed to define standards for multivendor network communications.
IT	Information Technology.
MAPI	Messaging Application Programming Interface. The messaging component of Microsoft's WOSA (Windows Open Services Architecture), which is built-in to NT Advanced Server.
MIME	Multipurpose Internet Mail Extension.
NFS	Network File System. NFS is a distributed file system from Sun Microsystems that allows data to be shared with many users in a network. NFS allows users to share data regardless of processor type, operating system, network architecture or protocol.
NTAS	Microsoft Windows NT Advanced Server.
OLE	Object Linking and Embedding. A feature of the Microsoft Windows operating environment and supported applications. Currently at revision 2 (OLE2).
ONC	Open Network Computing.
OS	Operating System.
OS/2	OS/2 is a single user, multi-tasking operating system developed by Microsoft and IBM that runs on 286-, 386-, and 486-based IBM compatible PCs.

OSF	Open Software Foundation.
OSI	Open Systems Interconnection. The OSI is a seven-layer communications reference model that has been defined by the International Standards Organization (ISO).
PEM	Privacy Enhanced Mail. An emerging standard for mail encryption.
PGP	Pretty Good Privacy. An emerging standard for Internet mail encryption.
POP	Post Office Protocol. An internet mail standard.
RIPEM	Riordan's Internet Privacy Enhanced Mail. An implementation of PEM.
RPC	Remote Procedure Call. A method used in service/client communications. Generally, the client issues the call as if it were a local procedure, and the service replies to the call, returning one or more parameters.
RSA	Rivest, Shamir, and Adelman. The names of the three developers who created the RSA encryption scheme.
SIG	Strategic Interest Group. A subset of NAC member companies focused on a particular strategic topic.
SMF 71	Simple Message Format. Novell's file-based interface to Global MHS.
SMTP	Simple Mail Transfer Protocol. The Internet standard protocol for transferring electronic mail messages from one machine to another. SMTP specifies how two mail systems interact and the format of control messages they exchange to transfer mail.
SNA	System Network Architecture. The network architecture developed by IBM which defines the specific mechanism for interconnecting terminals, eripherals, hosts and applications in an IBM environment.
SNMP	Simple Network Management Protocol. The Internet standard protocol for gathering network management information.
SQL	Structured Query Language. SQL is a language designed originally developed by IBM to interrogate and process data in a relational database.
TCP/IP	Transmission Control Protocol/Internet Protocol. TCP/IP is a communications protocol that is designed to interconnect a wide variety of different computer equipment.
Unix	Unix is a multiuser, multitasking operating system from AT&T that runs on a variety of computer systems from micro to mainframe.
VIM	Vendor-independent messaging. Formerly known as OMI (Open Messaging Interface, originally just Lotus and IBM). VIM committee

	now includes Lotus, Apple, Borland, IBM, MCI, Novell, Oracle, and Wordperfect. At one time considered a director competitor of MAPI, VIM has now been correctly positioned as an interface for add-on applications that run in conjunction with Notes and cc:Mail.
VINES	VItural NEtworking System. Network operating system from Banyan Systems, Inc.
WAN	Wide area network.
X.400	A CCITT and OSI specification for message exchange.
X.500	A CCITT and OSI specification for a hierarchical global directory.
XAPIA	X.400 API Association.