NAC POSITION PAPER

ENTERPRISE DIRECTORY SERVICES INTEGRATION

NETWORK APPLICATIONS CONSORTIUM
AUGUST 24, 1995

ABOUT NAC

The Network Applications Consortium (NAC) is a strategic organization dedicated to increasing the availability, quality and interoperability 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.

NAC members include:

ABB Power T&D Co.
American Bureau of Shipping
Arizona Public Service Company
Australian Bureau of Statistics
Banyan Systems Inc.
Bell Atlantic Mobile Systems, Inc.
Canadian National Railway
Carolina Power & Light Co.
Compaq Computer Corporation
Continental Grain Company
International Finance Corp., World Bank
MCI Telecommunications
Nike, Inc.
NYNEX
Pacific Gas & Electric
Pennsylvania Blue Shield
Public Service Electric & Gas
Resolution Trust Corporation
Tektronix, Inc.
Texaco
United States Marine Corps.
University of Michigan

This position 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 paper:

Continental Grain Company — *Hilly Fuchs*
MCI Telecommunications — *Brian Plackis*
MCI Telecommunications — *Gerry Robinson*
NetResults — *Doug Obeid*
Pacific Bell — *James Brentano*
University of Michigan — *Tim Howes*
University of Michigan — *Larry Gauthier, SIG Leader*
Editors — *Joe Sciallo and Kelli Wiseth, NetResults*

We welcome your feedback about this paper. For more information about NAC papers, contact:

Doug Obeid, Executive Director
Network Applications Consortium
c/o NetResults
5214-F Diamond Heights Blvd.
Suite 705
San Francisco, CA 94131

Copyright © August 1995 by NAC

Contents

Interoperability: Enterprise Directory Services

Executive Summary

Today's business environment is fast-paced and constantly changing. To be competitive in a global marketplace, corporations and their strategic partners must conduct business in the most cost-effective manner possible. Ideally, corporations should be able to leverage technology to facilitate this process. But conducting business between companies — or within a single corporation — brings with it a quagmire of problems. For example, it is difficult to find the addresses of people you need to send electronic mail to or network services you need to access. The solution to this problem is an integrated directory service that supports these functions and more. A directory service which houses names of users, services, and network applications is a key enabler for distributed computing — a relatively new concept in the industry.

What we have instead is a proliferation of application-specific directories — one for email, another for groupware applications, a third and fourth for corporate databases, and so on. If the directories share information at all, it is through a hodge-podge of gateways and other mechanisms, each of which must be installed, configured, and managed separately. Such an approach adds to costs and prevents the corporation from effectively leveraging directory information from one end of the enterprise to the other.

Directory services must be integrated within the organization and beyond, on a global scale. Integrating directory services within the organization leverages directory service functions and information while at the same time containing or reducing the administrative and support costs. Integrating directories on a global scale enables organizations to engage in communications and conduct business with each other as effectively as possible.

In this paper, NAC looks at directory service business drivers, summarizes the key issues, and makes specific recommendations to consumers and vendors. Each of these categories is summarized below.

Business Drivers

Organizations need to:

- Exploit information throughout the organization.
- Contain infrastructure costs (including initial acquisition, implementation, management, administration, and support).
- Support a geographically dispersed and mobile workforce.
- Communicate with business partners and customers on a 24-hour-a-day basis.

Benefits of Integrated Directory Services

Integrated directory services:

- Empower end-users to be more productive by enabling them to find what they need to do their jobs from anywhere, at anytime.
- Contain overall network cost-of-ownership by unifying directory services into a cohesive entity which other common services (security, messaging, and so forth) can leverage.
- Lower acquisition, deployment, management, and administration costs.
- Enable cost-effective deployment of distributed client-server applications in a global, mobile workplace.

Key Issues

- Vendors implement proprietary directories in their products. The result is that, in any given organization, a plethora of directories exists: one for each email system, one for each network operating system, one for each database application, and so on—and these directories do not interoperate to provide a unified and easily administered organization-wide directory service.
- Interoperability among directory services requires:
  - — a common application programming interface (API) between client desktop applications and the client directory service process. (See Figure 2 on page 8.)
  - — common protocols by which both client-to-service and service-to-service processes can communicate. (See Figure 3 on page 9.)
- Vendors haven't been given a clear vision of why they should implement standards in the directory services area, thus they see no clear incentive to adopt one.
- The popular network operating systems are only beginning to include a directory service implementation as a core component.

NAC's Recommendations

To Directory Service and NOS Vendors

- Design your directory with the ability to support inter- and intra-enterprise, global directory services. No matter how much market share you have, recognize that you will not be the only directory service vendor. Anticipate that your product must be able to interoperate with other vendors' products. (See Figure 1 on page 7.)
- Support the LDAP protocol (Internet RFC 1777) in your directory service to enable interoperability between any LDAP-compliant client and your directory service.
- Incorporate X.500's DAP and DSP protocols into your products to support directory-service-to-directory-service interoperability.

- Publish your directory service APIs and make them available to the industry. For example, Banyan's release of Universal StreetTalk (a software developer's kit for a non-VINES-specific version of its StreetTalk directory service API) is an approach that NAC endorses, especially because the API is license- and royalty-free, which makes it attractive to developers who might otherwise be tempted to build their own directories.
- Wherever relevant standards exist, adopt them. Participate in the IETF process model for standards development (in contrast to the OSF process model).[1]
- Participate in NAC's DIR (Directory Interoperability Rendezvous) to validate your directory service against a working X.500 implementation, modifying your product as appropriate for the sake of interoperability with X.500, and documenting the results.

To OS Vendors

- Incorporate the LDAP protocol (Internet RFC 1777) into your products to enable interoperability between client and LDAP-compliant directory services.
- Adopt a common desktop-application-to-directory-service API between the client and the directory service. Microsoft's proposed ODSI (Open Directory Services Interface) and Apple's AOCE (Apple Open Collaborative Environment) potentially provide the client APIs on the Windows and Macintosh desktops. NAC endorses both these desktop architectural models. (See Figure 5 on page 14.)
- Develop a common desktop application to directory service API (in the same vein as ODSI and AOCE) for IBM OS/2 and Unix client operating systems (Solaris, OSF Motif, HP Apollo, IBM, and others).

To Application Vendors

- ***Do not create proprietary directory services*** in your products.
  Rather, work with directory service vendors who offer open, readily available directory service APIs, and encourage directory services vendors to agree upon a common set.
- Use the available OS-level APIs.
  NAC supports Microsoft's proposed ODSI in the Windows environment and AOCE in the Macintosh environment in client applications.

To NAC, its Members, and Consumers

---

[1] Other models of productive standards development processes can be drawn from the hardware community. For example, the PCI (Peripheral Component Interface) bus standard, originally developed by Intel, has been adopted by virtually all desktop hardware vendors (PC, Mac, IBM, and Unix hardware). Further development of the standard is being shepherded by the vendor community at-large through the PCI-SIG, an industry-wide organization. In much the same manner as these hardware vendors, directory services vendors should agree upon a common set of APIs and compete on other factors (price, features, support, and so forth).

- Evangelize the benefits of directory services to the industry and facilitate the educational process by distributing the Directory Services Integration paper to business managers, colleagues, and vendors.
- Sponsor the DIR and actively encourage vendor participation.
- Assist directory services vendors in focusing their third-party development efforts. Act as catalyst and matchmaker to bring vendors and third-party developers together.
- Work with vendors to develop an effective business case for interoperable directory services.
- Participate in the IETF directory services working group activities.
- Leverage the work of the directory SIG into that of the security SIG to ensure a consistent message from NAC regarding interoperability and the common services model.

Note: Directory services depend on integrated security services, which are beyond the scope of this document. They will be covered by the NAC Security SIG in a future document.

Introduction

Scaleable, robust, interoperable directory services are the linchpin for enterprise-wide computing and for communications on a global scale.

In a global business environment marked by accelerated change and increased competition, the need for a flexible, integrated information technology infrastructure has emerged as the highest overarching priority for IT managers. The infrastructure must be based on interoperable components and common services that enable a corporation to leverage its investment in hardware, software, and information. Indeed, the pursuit of interoperability has been NAC's hallmark since the organization began in 1990.

Directory services are one of the most critical components of today's enterprise-wide information technology infrastructure. Directory services provide two key elements in the network computing environment, enabling:

- Name-to-address mapping between the name of a network resource and the low-level computer-oriented network name for that resource;
- people and resources to look-up other people or resources.

Both functions are necessary components of a flexible, dynamic information technology infrastructure. Name-to-address mapping enables a client process to find a server process; it is the basic glue that binds the network together. At a higher level, a lookup facility empowers people by giving them a unified view of all network resources.

However, a cohesive, integrated directory that serves both roles across the organization is the exception rather than the rule. Indeed, according to some analysts, only 15 percent of all corporations have implemented a directory service on their enterprise network today. That's because few network operating systems include a directory service as a core component. Instead, most vendors have historically provided a basic naming service, which serves the purpose of translating names into the network names used by network services.

Although this fundamental role is a very important one, name-to-address mapping alone provides a corporation no leverage when it comes to integrating other services and functions. For example, unlike a simple naming service, a directory service is capable of storing a wealth of additional information about the names contained in it, whether those names are for people, file or print servers, database applications, or the names of shared documents.

Thus, at a high level, NAC's vision of an enterprise-wide directory service has the following features:

- The directory service is key to client-server applications. Without a directory service, distributed applications cannot be effectively deployed.
- The directory service provides names of all enterprise resources to all people (and applications) in such a way that the infrastructure is a cohesive whole. Individuals need to be able to log in from anywhere, send a message to anyone from anywhere, or access file and print services regardless of location.

- The directory service is supportable at the enterprise-wide level. For example, administrators should be able to easily manage the mechanisms that keep the directory service in synch and up-to-date.
- The directory service integrates with the desktop and desktop applications. For example, the desktop email client and scheduling client must be able to use the same directory service.
- To people, access to a directory service is user-friendly, if not completely transparent.
- The directory service integrates with other common services. For example, the directory service must be accessible by the message transport-and-store service, by the security service, and so on.

Although Banyan has provided a directory service for many years that provides much of the functionality required at the enterprise-wide level, Banyan's marketshare is such that developers haven't written applications that take advantage of the directory service. So even in the Banyan environment, while the VINES directory service is tightly integrated with the VINES mail system architecture, directories for non-Banyan applications may exist as well—one for a database application, one for a new groupware application, and so on. And each of these directories must be installed, configured, administered, and maintained separately. When individuals leave the company or move to new locations, their address and other key information must be changed in multiple locations.

On the other hand, many applications have been developed over the years that work in the NetWare environment. But NetWare's full-featured directory service, NDS (NetWare Directory Service) is a recent innovation, found only in its NetWare 4.x product. Installations of NetWare 2.x and 3.x (which do not implement a directory service) still comprise the largest portions of Novell's installed base. The initial implementation of NDS is not integrated with Novell's own mail service, MHS. The result is that duplicate directory information continues to exist.

Thus, although a directory service may exist, it may not be used if the vendor doesn't own enough seats in the marketplace to make it attractive to developers. Or the directory service doesn't exist, and developers must write their own. Or, the directory service is not integrated with other infrastructure-level components.

In this paper, NAC examines the critical issues relative to directory services. First, NAC presents a generic model of a basic directory service. Against this model NAC examines vendor offerings, in relationship to the vendor's marketshare and stated strategic direction. Several key issues emerge, including the need for industry-standard APIs (application programming interfaces) and the need for industry-standard protocols. NAC examines vendor offerings in light of industry standards, where standards exist, and makes general recommendations to both vendors and consumers in this context.

Enterprise Directory Services Architecture

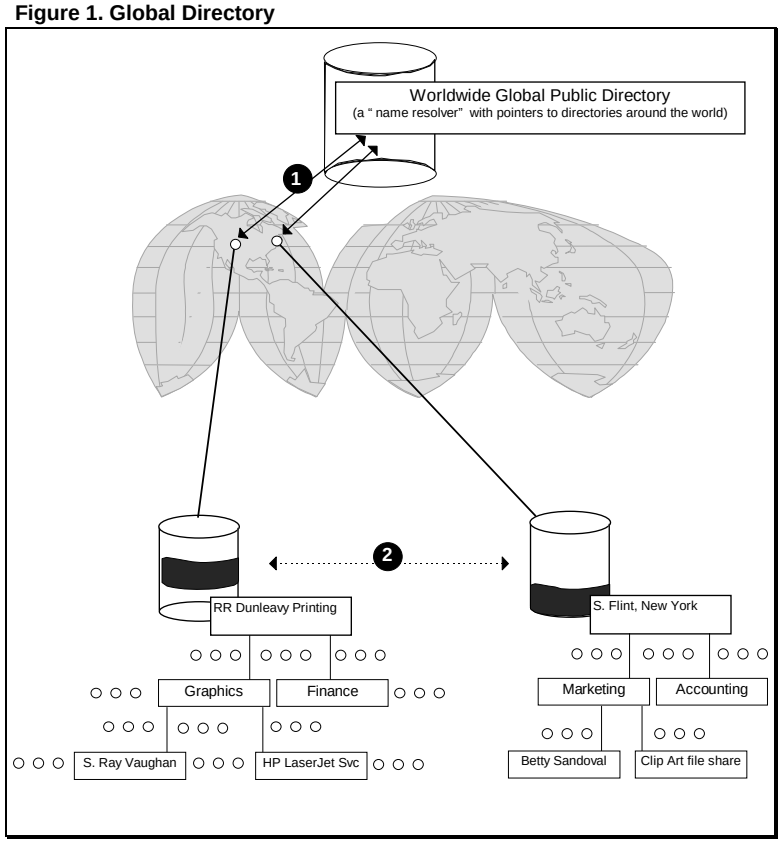
From the highest level vantage point, a global directory with worldwide, public access is "hierarchical" in nature and consists of all the local partitioned directories within countries, within various enterprises. The "directory" includes both the directory service (and the processes or functions that it performs) and the database of network objects to which the service provides access.

Figure 1 shows a conceptual view of the directories of two companies that might be doing business with each other at any given moment. In the figure, the database icons represent the master directory for each company; the dark portion of the directory represents private information that is kept within the confines of the corporation, but the remainder is that information which the company has decided to make publicly available. (Note that at the highest level, the worldwide public directory contains only public information.)

Figure 1 highlights these key concepts:

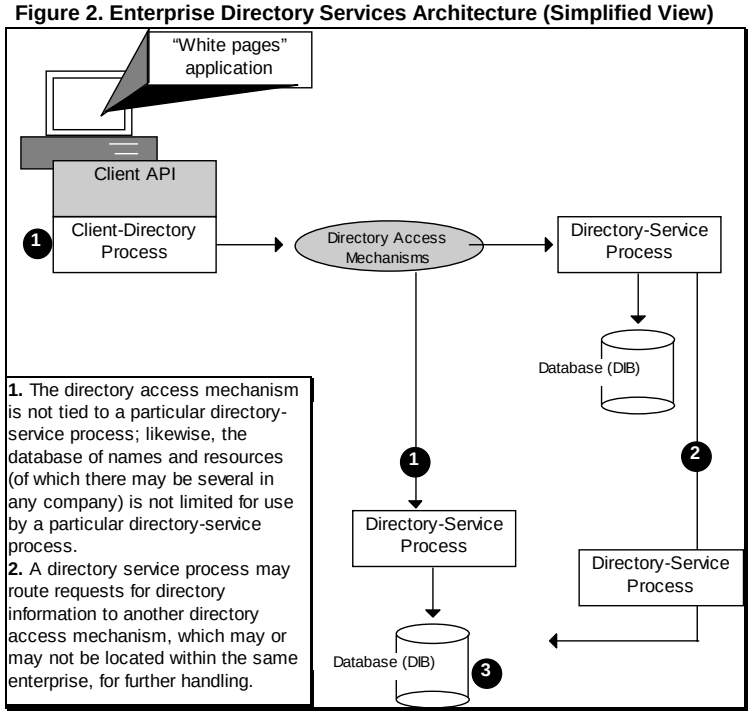
- The directory service stores names which can be structured however the organization sees fit, for example, along organizational lines, or as a single hierarchy.
- The directory within a given company functions as a "name resolver." If the named resource being sought is not available within the company, the directory service will: ❶ pass the request off to a "master name resolver" to locate the appropriate directory; and possibly, ❷ use the directory name resolver on an intra-company basis as well.

Figure 1. Global Directory



The structure of the database and its content is one perspective on directories; the processes and functions that the directory service performs is another. At a basic level, a directory service must be composed of the three key components depicted in Figure 2.

Figure 2. Enterprise Directory Services Architecture (Simplified View)



❶ A **client-directory process** that enables an individual or another client application to make use of the information housed in the database. Applications on the desktop share a common client API.

❷ A **directory-service process** that provides a variety of functions, including:

- naming network resources
- resolving name lookups and routing lookup requests to another directory service when necessary
- managing a database of network resources
- distributing information to other directory-service processes
- collecting information from other directory-service processes
- delivering information to client processes directly or via other directory-service processes

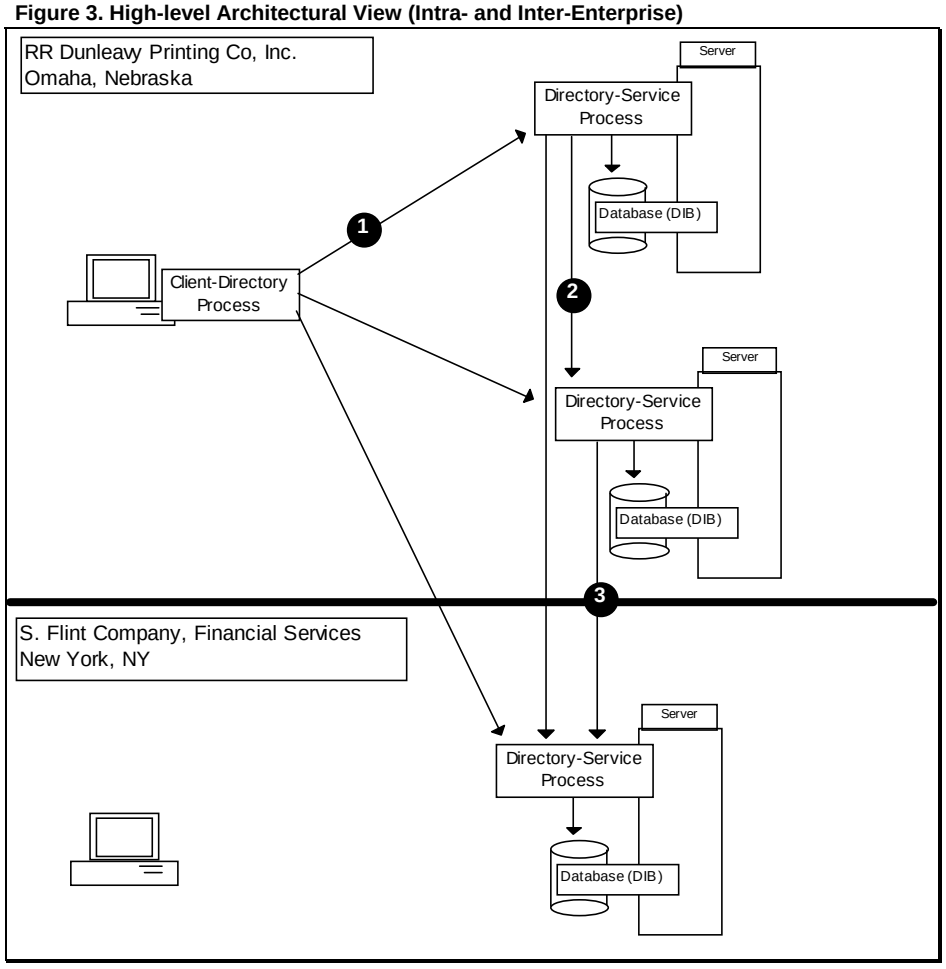
❸ A **database** that contains network resources, the objects, and attributes.

Connectivity between the client and directory service components is supported by an **access mechanism** which is not tied to any particular directory-service process.

Furthermore, the access mechanism cannot be not bound to the local level. As seen in Figure 3, the access mechanism must support access between a client and a directory

process: ❶ within the local level; ❷ on an **intra-enterprise** basis; and ❸ on an **inter-enterprise** basis.

Figure 3. High-level Architectural View (Intra- and Inter-Enterprise)



In addition to the three core components—client, directory service, and directory database—and the basic functionality that they provide, other basic common services must be provided; given that the directory operates in a distributed, network computing environment, there's the inherent requirement for security, for synchronized events, and the like. These common services include:

- A ***security service*** which *authenticates* users (verifies the identity of the requester), and, once authenticated, ensures that users have access to services and information for which they have been *authorized* to use. A security service may also provide encryption for even more secure network transactions.

- An ***event-stamping service*** which ensures the synchronization of events; the event-stamping mechanism provides an independent means to identify each event that occurs (add an object to the database, delete an object from the database, modify an object in the database, and so on). Time is often used as the event stamp, hence the event-stamping mechanism is also frequently referred to as the *time service.* Management functions, such as audit logging, also rely on event stamping.

- A ***management system*** that supports administration and management of users, services, and all other network entities. This is particularly important in an enterprise environment where there may be thousands of users and dozens, if not hundreds, of services.

Each component in this model is a functional unit with separate interface points. These key interfaces are discussed in the next section.

Note: In the NAC model, the directory service process itself may provide some of the functionality of security, time-stamping, and management services, or an independent service running elsewhere in the enterprise may be called by the directory-service process to perform these functions. Refer to Appendix B for more information.

Enterprise Directory Service APIs and Protocols

An analysis of the interface points provides distinct sets of functions that enable the enterprise directory service.

Overview

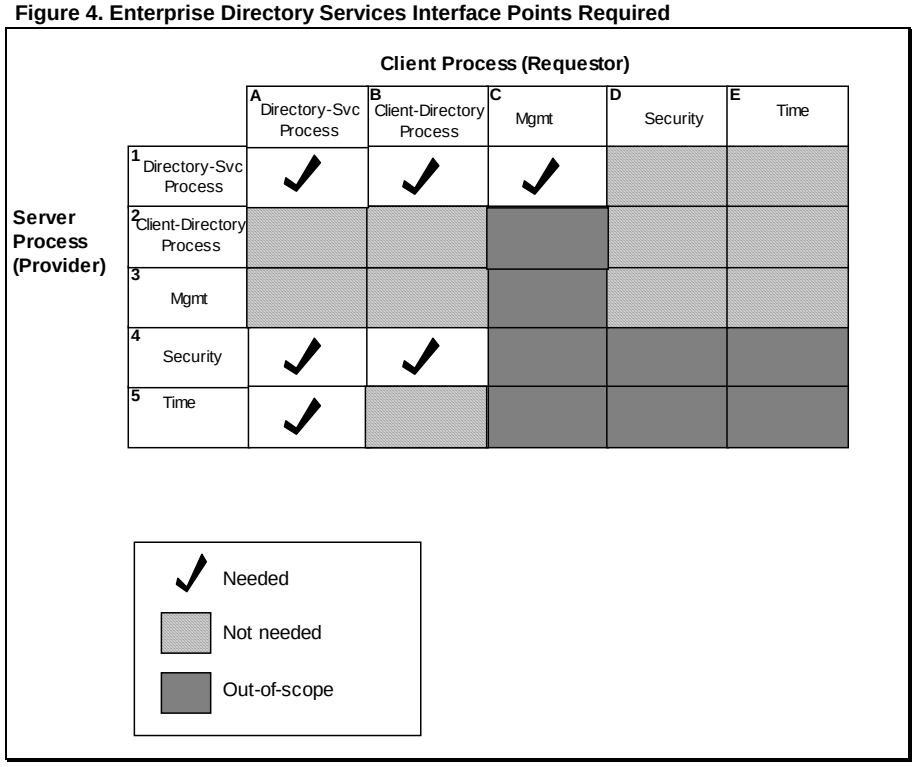
An API provides a level of functionality and application support for the operating system. One way to navigate seamlessly through different directories is to come up with a common, standard directory API. Developers can write to a single API, rather than distinct interfaces for each vendor's directory. Not only does this simplify matters for users and administrators, it also results in developers writing more directory-enabled applications. A recent example of how this holds true is the development of the Windows Socket API (Winsock) for running Windows applications over any vendor's TCP/IP protocol stack. Windows applications utilizing the Winsock API are almost as common these days as word processing programs.

A protocol is the body of rules that enables the orderly, reliable transfer of data among nodes (clients and servers) on a network. Typically, in the context of the International Standards Organization (ISO) seven-layer model, a protocol refers to the rules associated with a specific layer or set of layers. At any layer, the protocol includes standard interfaces for requesting service from the layer below, and for providing service to the layer above. Directory services use a variety of protocols, between clients and directory services, and between directory services themselves. Because no standard existed when directories were initially being developed, each vendor of a directory service, including Banyan, Novell, and Microsoft, developed their own client-to-service and service-to-service directory protocols.

Interface Matrix

Taken together, APIs and protocols represent "interface points" at which different components of an enterprise system, such as a directory service, need to interoperate. Figure 4 on page 12 represents a matrix of the six basic interface points required for an enterprise directory service.

Figure 4. Enterprise Directory Services Interface Points Required



To summarize, the critical points in terms of interoperability among different vendor's products, and in terms of integrating various enterprises into a seamless whole, are:

A1: Directory service process to directory service process

B1: Client directory process to directory service process

Both of these interfaces potentially require a common API and protocol to enable interoperability.

Note: In conjunction with developing this matrix, NAC has developed a generic listing of the functions that must be provided at each interface. 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. Refer to Appendix C for this listing.

Functional Analysis

Directory APIs

NAC believes that a common API is needed for client applications on the desktop to comunicate with the directory service.

Other supporting APIs are necessary to facilitate communications between:

- Directory service and security service
- Management service and directory
- Client application and security service
- Client application and directory service manager

These APIs are outside the scope of this paper. It is not that they are not required, or are somehow less important. The focus of this paper is on the client-to-directory, and directory-to-directory access. NAC recognizes, for instance, that security and management are important issues when discussing directory services, but believes that these issues are best left to another discussion.

Directory Protocols

NAC believes that standard directory protocols are necessary to enable communications between:

- Client directory process and directory service process
- Directory service process to directory service process, both within the same domain (intra-directory communication) and between different domains (inter-directory communication)

In addition, NAC believes that a **standard namespace** (or schema) is necessary, because it determines much of the functionality of the directory. Since the namespace establishes what objects are in the directory, and how they are named, it has a great influence on directory interoperability. It is one thing to be able to provide access to a directory, and another to be able to correctly interpret let alone display information contained in another directory's namespace implementation. Agreeing to a base-line standard schema is necessary to define basic objects and attributes common to all directories, and make interoperability more of a reality.

Competing Implementations of the API and Protocol Sets

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

Figure 5 on page 14 displays some of the proprietary and standard interfaces and protocols currently implemented or planned in directory 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 5. Functional API and Protocol Sets

	Directory	Client APIs	Client-Directory Protocols	Directory-Directory Protocols
Apple	PowerShare Catalog	AOCE	AppleTalk	AppleTalk
Banyan	StreetTalk (STDA) (Universal StreetTalk)	VINES API (DAPI)	VINES (LDAP, DAP)	VINES (DSP)
IBM	CDS/GDS	NSI, XDS (XFN)		GDA
Lotus/ SoftSwitch	cc:Mail Notes EMS	VIM (MAPI, CMC) Notes		
Internet	DNS MX	POP, MH		
Microsoft	NT Domain MS Mail	NT API MAPI, OLE (CMD) (ODSI)		
Novell/ WordPerfect	NetWare NDS, Global MHS Directory GroupWise Directory	NCP, SMF71, AOCE, MAPI, GroupWise XTD	NCP	NCP
OSF	CDS/GDS	X/Open, NSI, XDS (XFN)		GDA
OSI	X.500	XDS/XOM	LDAP, DAP	DSP
SunSoft	NIS+, NIS (Federated Naming)	XFN		

Note: The items in parenthesis indicate either planned support or an emerging architecture.

Vendor Evaluations

Most, if not all directory services, including Novell's NDS and Banyan's StreetTalk, provide APIs for application developers. The problem is that developers need a single standard API to directory services. They need to be able to construct applications that work with a variety of directory services on a variety of networks. The other problem with today's applications utilizing directory services is that vendors are promoting implementation-specific APIs to those services.

Furthermore, the leading vendors have not acknowledged or ignored the fact that the predominant client operating system controls the desktop market. Since Windows is the most widely used client operating system, it seems inevitable that APIs from Microsoft should dominate. It is not until very recently that Microsoft announced plans to provide access points to directory services that are natural extensions to Windows. Called Open Directory Services Interfaces (ODSI), these APIS—of which at least four are promised—are service providers and isolation layers that will be part of the operating system. Without widespread support for these APIs, NAC believes this will continue to result in a shortage of willing developers to provide directory-enabled applications.

From the standpoint of directory protocols, the story is much the same. While a "standard" was being developed, vendors were bringing directory offerings to market to meet customer demand. (In fact, the 1988 X.500 standard, which included DAP and DSP protocol specifications, only reached International Standard status in 1990.) Because no standard directory protocols existed at the time, and because the "standards" process is so slow, directory service vendors had to provide their own client-to-directory, and possibly directory-to-directory protocols. The result is a lack of commonality at this level as well.

The next sections summarize each vendor's or standards body's products and architectural direction, illustrating the variety of directory implementations and planned implementations in the market today.

Apple

The Apple Open Collaboration Environment (AOCE) is part of the Macintosh operating system, providing modular messaging, directory, and authentication services, and APIs for each service. Server Access Modules (SAMs), (which function similarly to Server Provider Interfaces, or SPIs) are also included, permitting access to external databases and messaging systems. This would enable, for example, an Apple mail front-end to access a non-Apple directory.

While AOCE's architecture is in line with NAC's direction, to date it has been limited to the Apple environment.

Banyan

In the directory services arena, many analysts agree that Banyan is the vendor with the best directory track record in the market. The problem, which is not unique to Banyan, is that this is a proprietary directory. Recently, Banyan announced plans to make StreetTalk a standard with its DAPI (Directory Application Programming Interface) and Universal StreetTalk initiative. Under the plan—code-named "Redwood"—Banyan gives away licenses and APIs of StreetTalk to software vendors, encouraging vendors to construct applications around it. It is also likely that StreetTalk will run on more platforms in the near future, including Windows NT.

Like AOCE, Banyan's announced architecture for StreetTalk supports the WOSA model, enabling users to plug in the directory of their choice on the backend. While NAC applauds this development, it is reasonable to question whether Banyan has the clout to make StreetTalk an industry standard.

Banyan recently announced support for the Microsoft ODSI APIs (see Microsoft, page 17), as well as the LDAP, DAP, and DSP protocols. NAC views this as a positive step forward in making directory interoperability more of a realistic achievement.

IBM-Lotus

IBM's recent acquisition of Lotus certainly holds the promise of many changes, but also makes it uncertain as to how and when directory service products will be forthcoming from this new merger. It is thus helpful, in the meantime, to explore what each company's direction has been so far.

IBM

In short, IBM's vision—the "Open Blueprint" architecture—is to be a supplier of distributed systems. The IBM environment is a group of mainframes, mini computers, and Unix servers, all tied together by the Distributed Computing Environment's (DCE) services. IBM believes that DCE is both good technology and a good environment, because that technology is "open." The problem with this vision is that it is not relevant to the PC space.

Recently, IBM has been touting a global DCE directory (currently in beta testing at the time of this publication). This directory is based on a standard set of services such as directory, security, and application development calls, that theoretically will run across all platforms one day. It will be part of "Extended Feature," an add-on to LAN Server 4.0, slated to ship by the end of 1995. According to IBM, the directory includes a graphical user interface which shields the administrator from the complexity of the DCE directory. As part of DCE, the directory is also promised to run on Unix and mainframe platforms.

Nevertheless, the fact remains that DCE has not exactly taken off yet and remains a tough sell to small to medium-sized business environments. These organizations are typically running Windows and DOS, for which DCE implementations create a huge and impractical overhead. No real off-the-shelf applications yet exist for DCE, further hampering its acceptance.

On the topic of client side APIs, IBM acknowledges their necessity, but falls somewhat short in being able to convey to developers what APIs should be written to in the IBM/DCE environment. IBM has yet to even say that it's committed to providing an SPI architecture for OS/2, much less define APIs, publish specifications, and deliver SDKs.

Lotus

Lotus acknowledges that it does bring to the market many directories: cc:Mail, Notes, and so on; and that this is confusing to developers and users alike. It has not stated if and why it will continue to use all these directories, one of them, or some other vendor's directory. In short, Lotus has yet to articulate a single directory strategy, or integrate its existing directories. The area where Lotus seems strongest is that of directory synchronization. To Lotus, directory synchronization does work, and it is committed to keep on improving its products in this area.

Transitioning from a provider of desktop applications, Lotus was been pulled into the enterprise network services arena. The Lotus/SoftSwitch union of 1994 gave Lotus a combination of messaging and information-sharing applications (cc:Mail and Notes); key application services such as replication (Notes), directory (LCS), and message store (LCS); and message backbone services (SoftSwitch EMX). While full of possibilities, Lotus has yet to establish a clear strategy amidst all of these product offerings.

The fact that Lotus announced support for Microsoft's ODSI APIs is seen as a positive step towards resolving vendor API wars and providing users with standard interfaces they need. To go one step farther, Lotus should support other companies' directories.

Internet

While DNS (Domain Name Service) is widely installed, enabling many millions of hosts on the Internet to locate one another, this directory service does not extend beyond name lookup. It does not provide or enable any other functionality that people have come to expect of an enterprise directory (for example, allowing applications to locate one another). DNS has all the earmarks of a Unix-based open standard which has not been ported to the PC market. As such, it will continue to fill its niche role for Internet/Unix communications.

Microsoft

Microsoft is aiming to make major inroads as a network services provider. In this respect, it has a long way to go, but at least it is articulating a plan on how to get there while slowly providing most of the necessary software and applications.

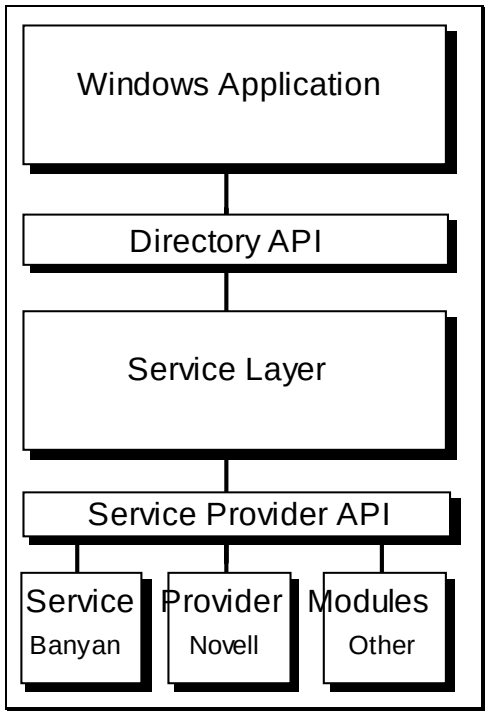
The problem is that its flagship product, Windows NT Server, doesn't have a directory service, and to be competitive in enterprise networking, a robust directory service is essential. What Microsoft has instead is its antiquated domain service, which has a few of the aspects of a directory service (so it provides basic name-to-address mapping), but unfortunately misses on most important counts.

Novell is quick to point out that NDS is much more extensible than NT Domain Service, and this is an accurate statement. NT Domain may be adequate for email and day-to-day NOS administration, but for true enterprise applications, it falls short of the needs of distributed computing.

In the world of interoperability between it and its rival, Novell, Microsoft wants NT Servers to be able to control NDS in the enterprise. Not agreeable to this endeavor, Novell wants that role to remain with NetWare servers only, and to let NT Servers be visible to NDS, but not control the directory service. This argument makes little sense. It remains to be seen what, if anything, will be able to be worked out on this front.

What Microsoft does have is the Windows Open Services Architecture (WOSA). WOSA is a set of isolation layer APIs for network service access, including Windows Sockets (a transport independent API), MAPI (for messaging), and TAPI (for telephony). These APIs are architected as a Service Provider Interface (SPI), as depicted in Figure 6 on page 18.

Figure 6. Service Provider Interface Model



Essentially, the operating system provides an API to a given type of service, that is, to directory services. This API "generalizes" a baseline set of functions. In the case of a hypothetical directory API, such baseline functions would be a basic set of objects, such as user, printer, file, and so on, and then functions that could be performed on those objects, such as accessing an object, reading and/or creating attributes of that object, creating objects and so on. These functions are "generalized" so that any application can use them in a consistent fashion via the API.

The service layer maps the "generalized" functions of the OS-level API to a lower-level API called the service provider interface. Service vendors, such as Banyan, Novell, WorldTalk, and so on, write service provider modules that support their products to the service provider interface. The generalized calls relayed to the SPI by the service layer are mapped to implementation specific calls by the service provider module.

Vendors should make their SPI modules freely available for no charge. Then, application developers may want to include the module with their application, and as part of the installation process, make sure the appropriate SPI is installed so the application works.

In July of this year, Microsoft presented its Open Directory Services Interfaces (ODSI), which are service providers and isolation layers that will be part of the NT and Windows95 operating systems. So far, four ODSI APIs are forthcoming:

- **Network Provider Interface**—enables a single login-like capability for multiple network operating systems and directories.
- **WinSock (RnR) and RPC**—registers services to the directory and provides for lookup browsing.
- **OLE DB**—by bringing OLE services to the directory, complex queries on the directory can be developed and run.
- **OLE DS**—works with schema management.

While some might feel that the Microsoft ODSI effort is a way to "save face" until the advent of its enterprise directory code-named Cairo, NAC supports these APIs in the Windows environment as a way to get application developers to create applications that will work across directory implementations.

Novell/WordPerfect

With the release of NetWare 4.x, Novell has jumped into the NOS directory service arena, and is providing more than just its server-centric, domain implementation knows as the Bindery. Novell has announced plans to run NDS on many other platforms, such as Windows NT and OS/2, providing a single login and administration point in heterogeneous networks. Novell publishes the NetWare 4.x NWS (NetWare Directory Service) API NLM, which is its toolkit for developers who wish to write applications that can take advantage of the NetWare Directory Service.

Acceptance of NetWare 4.x is already the largest enterprise directory in the market today. Like the other major directory players, Novell wants to make NDS the industry standard directory. Not wanting to concede anything to Microsoft, Novell is developing a version of NDS for Windows NT. While this strategy seeks gain the lion's share of the directory market, it does not move Novell any closer to interoperating with existing directories.

It is unclear at this time if Novell will pledge support for Microsoft's ODSI APIs, since it appears to believe that control of the API is tantamount to control of the directory market itself. To date, this has been one of the main problems with Novell's vision: lack of support for APIs outside of its own. NAC believes that Novell should follow suit with Banyan and Lotus in pledging support for the ODSI APIs. Such support could actually increase Novell's rollout of NDS by enabling applications that are NDS-aware. Additionally, NAC would like to see Novell announce support for X.500 protocols, such as LDAP, DAP, and DSP.

OSF

The Open Software Foundation (OSF) is a consortium of companies that have been cooperating on developing open systems software known collectively as Distributed Computing Environment (DCE). This consortium functions as follows. Once OSF decides on a technology, it asks its members to submit applications to develop the software. As the company, under OSF guidance, produces the software, OSF in turn licenses the DCE software. The other member companies port the code to their platforms, sell the product, and provide royalties to OSF. OSF then puts a percentage of that money back into the companies coding and providing the software.

To date, there has been no widespread acceptance of DCE products, though many vendors have lined up behind the OSF banner. One key reason is that OSF has completely ignored the PC space. Another reason is that the cost for entry into DCE is high, both in terms of dollars and in terms of processing support required to run the code modules. Furthermore, DCE is very Unix-oriented. Only a year ago was a DCE development kit for the Windows desktop released. Again, developers are attracted to writing code for products that enjoy large marketshare, and without a viable Windows DCE component, acceptance has been slow. And DCE is a top-to-bottom approach.

SunSoft

Until recently, SunSoft has confined its directory offerings to the Unix namespace, providing a domain-like directory called the Name Information Space, or NIS+ (the first iteration was known simply as NIS). NIS+ is a component of the Open Network Computing (ONC+) environment, which is a family of distributed computing services. While offering many benefits one would expect from a directory service—simplifying network administration, replication, authentication, easier access to network resources—NIS+ is not a full-fledged directory service. It is not a general purpose, enterprise directory, capable of storing large and complex amounts of inter-organizational data.

SunSoft's strategy for providing interoperability for multiple namespaces is to provide two interfaces: the Name Service Switch, which only provides compatibility between NIS, NIS+, and DNS; and federated naming.

Federated Naming is a more extensive solution to namespace interoperability. It is supposed to support multivendor systems, and provide global and enterprise naming services, in a "plug-and-play" fashion. Thus, in this system, NetWare, DCE, OSI, NIS+, and more, should be able to not only coexists but interoperate.

SunSoft is currently at work on an overall distributed computing architecture called Spring. This operating system would automatically create the distributed services architecture necessary for client-server computing, and would include modular services such as directory and security. While promising, it remains to be seen when such a product can be brought to the table, and what impact if any it will have outside of the Unix environment.

X.500

The X.500 suite of standards is not a panacea; instead of being *the* solution, it will be only a *part* of the solution. NAC believes these parts include LDAP, DSP, and DAP (on the server). Important factors that are beyond the scope of X.500—such as application development frameworks, integration of directories with distributed file systems, the need for practical architectures, and interoperability with existing directories—will influence directory implementation as well.

X.500 will be a factor, and the fact remains that all of these issues are relevant to the directory discussion. For example, X.500 and other standards will have a definite impact on the development of directory services, particularly in the way that a company's internal systems interact with external systems, both public and private.

Furthermore, most analysts agree that native X.500 carries too much overhead for implementation on Windows and DOS clients, which form the bulk of corporate installations today.

What X.500 does bring to the table is a set of protocols that NAC believes can and should be adopted as standards by the industry for directory-to-directory communications. Though one of the main problems with X.500 is that it is specification rather than implementable code, it does contain pieces that vendors could agree upon and adopt as common components enabling the kind of interoperability NAC seeks to achieve.

NAC Recommendations

A summary of recommendations for vendors, consumers, and NAC members.

To Directory Service and NOS Vendors

- Design your directory with the ability to support inter- and intra-enterprise, global directory services. No matter how much market share you have, recognize that you will not be the only directory service vendor. Anticipate that your product must be able to interoperate with other vendors' products.
- Support the LDAP protocol (Internet RFC 1777) in your directory service to enable interoperability between any LDAP-compliant client and your directory service.
- Incorporate X.500's DAP and DSP protocols into your products to support directory-service-to-directory-service interoperability.
- Publish your directory service APIs and make them available to the industry. For example, Banyan's release of Universal StreetTalk (a software developer's kit for a non-VINES-specific version of its StreetTalk directory service API) is an approach that NAC endorses, especially because the API is license- and royalty-free, which makes it attractive to developers who might otherwise be tempted to build their own directories.
- Wherever relevant standards exist, adopt them. Participate in the IETF process model for standards development (in contrast to the OSF process model).
- Participate in NAC's DIR (Directory Interoperability Rendezvous) to validate your directory service against a working X.500 implementation, modifying your product as appropriate for the sake of interoperability with X.500, and documenting the results.

To OS Vendors

- Incorporate the LDAP protocol (Internet RFC 1777) into your products to enable interoperability between client and LDAP-compliant directory services.
- Adopt a common desktop-application-to-directory-service API between the client and the directory service. Microsoft's proposed ODSI (Open Directory Services Interface) and Apple's AOCE (Apple Open Collaborative Environment) potentially provide the client APIs on the Windows and Macintosh desktops. NAC endorses both these desktop architectural models.
- Develop a common desktop application to directory service API (in the same vein as ODSI and AOCE) for IBM OS/2 and Unix client operating systems (Solaris, OSF Motif, HP Apollo, IBM, and others).

To Application Vendors

- ***Do not create proprietary directory services*** in your products.
  Rather, work with directory service vendors who offer open, readily available directory service APIs, and encourage directory services vendors to agree upon a common set.
- Use the available OS-level APIs.
  NAC supports Microsoft's proposed ODSI in the Windows environment and AOCE in the Macintosh environment in client applications.

To NAC, its Members, and Consumers

- Evangelize the benefits of directory services to the industry and facilitate the educational process by distributing the Directory Services Integration paper to business managers, colleagues, and vendors.
- Sponsor the DIR and actively encourage vendor participation.
- Assist directory services vendors in focusing their third-party development efforts. Act as catalyst and matchmaker to bring vendors and third-party developers together.
- Work with vendors to develop an effective business case for interoperable directory services.
- Participate in the IETF directory services working group activities.
- Leverage the work of the directory SIG into that of the security SIG to ensure a consistent message from NAC regarding interoperability and the common services model.

Note: Directory services depend on integrated security services, which are beyond the scope of this document. They will be covered by the NAC Security SIG in a future document.

Conclusion

It's clear that directory services are essential for distributed computing, but enterprise and global directory solutions are not going to be available in the short term. However, NAC is committed to moving the industry forward to resolve directory interoperability integration for inter- and intra-enterprise computing. NAC invites all of you to work with us in this endeavor.

Vendors should begin evolving their existing products to use a common desktop client-to-directory process API. In the Windows environment this is ODSI, and in the Mac world it is AOCE.

Vendors should also build in support for the LDAP protocol, for desktop client-to-directory process communications, and for the DAP and DSP directory protocols that enable directory-to-directory communications.

The reality is, in the short-term, many companies will have to rely on gateways and synchronization products to deliver directory interoperability. Nevertheless, these means should not be viewed as the end-all, be-all solution. They are band-aids only. Vendors need to move quickly toward supporting standard APIs and protocols.

References

AOCE Application Interfaces. Inside Macintosh. Apple Computer, Inc. 1994.

AOCE Service Access Modules. *Inside Macintosh*. Apple Computer, Inc. 1994.

Barker, P., Kille, S. *The COSINE and Internet X.500 Schema: Request for Comments 1274. (Category: Standards Track)*. Network Working Group. November 1991.

Barker, P., Kille, S., Lenggenhager, T. *Naming and Structuring Guidelines for X.500 Directory Pilots: Request for Comments 1617. (Category: Informational)*. Network Working Group. May 1994.

Black, Uyless. *Data Communications and Distributed Networks*. Prentice Hall, Englewood Cliffs, NJ. 1993.

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. *Directory Service Standards: X.500*. Burton Group Report. October 1992.

Burton, Craig and Lewis, Jamie. *Directory Services: Strategic Overview*. Burton Group Report. October 1992.

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

Chadwick, David. *Understanding X.500: The Directory.* Chapman & Hall, London. 1994.

Day, Michael; Koontz, Michael; and Marshall, Daniel. *Novell's Guide to NetWare 4.0 NLM Programming*. Novell Press, San Jose, CA. 1993.

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

Getchell, A., Sataluri, S. *A Revised Catalog of Available X.500 Implementations: Request for Comments 1632. (Category: Informational)*. Network Working Group. May 1994.

Jurg, P. *Introduction to White Pages Services based on X.500: Request for Comments 1684. (Category: Informational)*. Network Working Group. August 1994.

Khanna, Raman (Editor). *Distributed Computing: Implementation and Management Strategies*. Prentice Hall, Englewood Cliffs, NJ. 1994.

Kille, S. *Using the OSI Directory to Achieve User Friendly Naming: Request for Comments 1781. (Category: Standards Track)*. Network Working Group. March 1995.

Lewis, Jamie. *Strategic Overview: The Advent of Directory-Enable Computing*. Burton Group Report. July 1995.

Mansfield, G., Johannsen, T., Knopper, M. *Charting Networks in the X.500 Directory: RFC 1609. (Category: Experimental)*. Network Working Group. March 1994.

Mansfield, G., Kille, S. *X.500 Directory Monitoring MIB: Request for Comments 1567. (Category: Standards Track)*. Network Working Group. January 1994.

Postel, J. *Domain Name System Structure and Delegation: Request for Comments 1591 (Category: Informational).* Network Working Group. March 1994

Rose, M. *Directory Assistance Services: Request for Comments 1202*. Network Working Group. February 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.

Shirley, John; and Rosenberry, Ward. *Mircosoft RPC Programming Guide*. O'Reilly and Associates, Inc. Sebastopol, California, 1995.

Yeong, W.; Howes, T.; and Kille, S. *X.500 Lightweight Directory Access Protocol*: *Request for Comments 1777*. Network Working Group. March 1995.

Appendix A. Requirements Summary

The Network Applications Consortium began addressing issues relative to enterprise directory services in the winter and spring of 1995. NAC's Directory Services SIG (Strategic Interest Group) published a requirements document (available on the World-Wide Web at http://www.tbg.com/nac), the *Enterprise Directory Services Integration SIG: Requirements Paper*, which puts many of the marketing and technical issues on the table for discussion. This appendix reviews some of the basic concepts presented in that document first, with a summary of the requirements.

Overview of Naming, Objects, and Attributes

A network object is anything connected to the network, including people, file servers, print services, distributed databases, directory services, security services, and so on. In a networked, distributed computing environment, every one of these objects, or network resource, must have an address so that it can be located. That address can be a cryptic, computer-oriented name—139.121.14.25, for example—or it can be a name that makes sense to human beings, from file clerk to CEO—Marketing.File.Server, for example. Naming is a fundamental function in any and all network operating system environments, and it is a key function in NAC's architectural model.

In the model, the naming function is included as a core component of the directory service. This is the case in some network operating system environments, such as Banyan VINES and Novell NetWare 4.1; but in some cases, such as Windows NT Server 3.5, only the naming function is performed and a full directory service as envisioned here is not implemented at present.

By itself, naming isn't enough. Both people and processes need to access these names, and the purposes for which people and processes need a name vary. As the short list above reveals, objects (person, file servers, print services, distributed databases, directory services, security services, and so on) fall into different categories, and so they can be organized as such. Thus, objects are defined by class, or category. A user object has a different class type associated with it than, say, a database object.

Class type is just one of the *attributes* associated with an object. Depending upon the class type, other attributes associated with an object will vary as well. For example, a user object might have attributes such as "phone number, job title, salary, department name, supervisor name" and the like associated with it. Attributes associated with a database object will be very different from these, and might include items such as security levels, access privileges, and the like (although user objects contain such attributes as well.)

All objects and their attributes are maintained in a specialized database, often referred to as a "directory information base," or DIB. Like other databases, the DIB should be searchable by either the object name or by one of its associated attributes, on a class-wide basis. In English? "Find me Dan Clarke" (object name) or "Find me all the color laser printers on the 7th floor" (attribute search).

These fundamental notions and constructs, then, anticipate the requirements for the directory service and the NAC model which follows. The functional, qualitative, and

architectural requirements from the *Enterprise Directory Services Integration SIG: Requirements Paper* are summarized below.

Functional Requirements

The directory must:

- Provide name-to-address mapping which enables dynamic binding between a network object and its network location
- Be scalable to many people, resources, and locations
- Enable objects to be named using a multi-part naming structure
- Support aliases
- Be extensible so that the multi-part naming structure (the attributes) can be used to meet an organizations needs
- Provide for partitioning and the maintenance of private information within an autonomous network, while externally sharing the unified corporate directory
- Interoperate with other unified directories in other companies and nations around the world (address the need for global inter-namespace interaction)
- Notify registered foreign directory services as directory changes occur
- Be fault-tolerant
- Use event stamping to ensure the synchronization of events with the distributed/replicated databases
- Allow a user to provide incomplete information and then refine the search based on matches that occur
- Be able to lookup by categories or classes (that is, white page lookups on attributes) or by names of services on the network (yellow page browses)
- Support very specific security requirements, including: C2 security at a minimum, with B2 assurance. B2 assurance requires vendors to convincingly demonstrate that:
  - the applications (and users) are protected from each other
  - the operating system is protected from the users
  - the operating system performs as designed under all conditions
  - use the network authentication, access control and rights, and it must support adequate access controls
  - be modular and flexible enough to accommodate any specific security model
  - single-entry of user information between the directory and security services

Qualitative Requirements

An enterprise directory service must be:

- Based on a naming scheme that is readable by humans ("nancy_peterson@macuser.ziff.com" vs. "139.111.23.15," for example)
- Highly available and provide quick access to data
- Robust enough to provide consistent lookup performance and directory accessibility
- Able to handle bursty access consistently so that performance is always acceptable
- Able to be managed through an integrated platform
- Able to be administered through an integrated application

Architectural Requirements

The functional and qualitative requirements listed above lead to some specifics in terms of system design. The directory must use:

- A distributed architecture, that is, the directory service and the database of names must be distributed throughout the network to achieve the optimal performance and reliability demanded in an enterprise-wide environment;
- Quality checking mechanisms and synchronization schemes, and these mechanisms must be easy to implement and manage from an administrative standpoint.

Appendix B. Directory Service Process

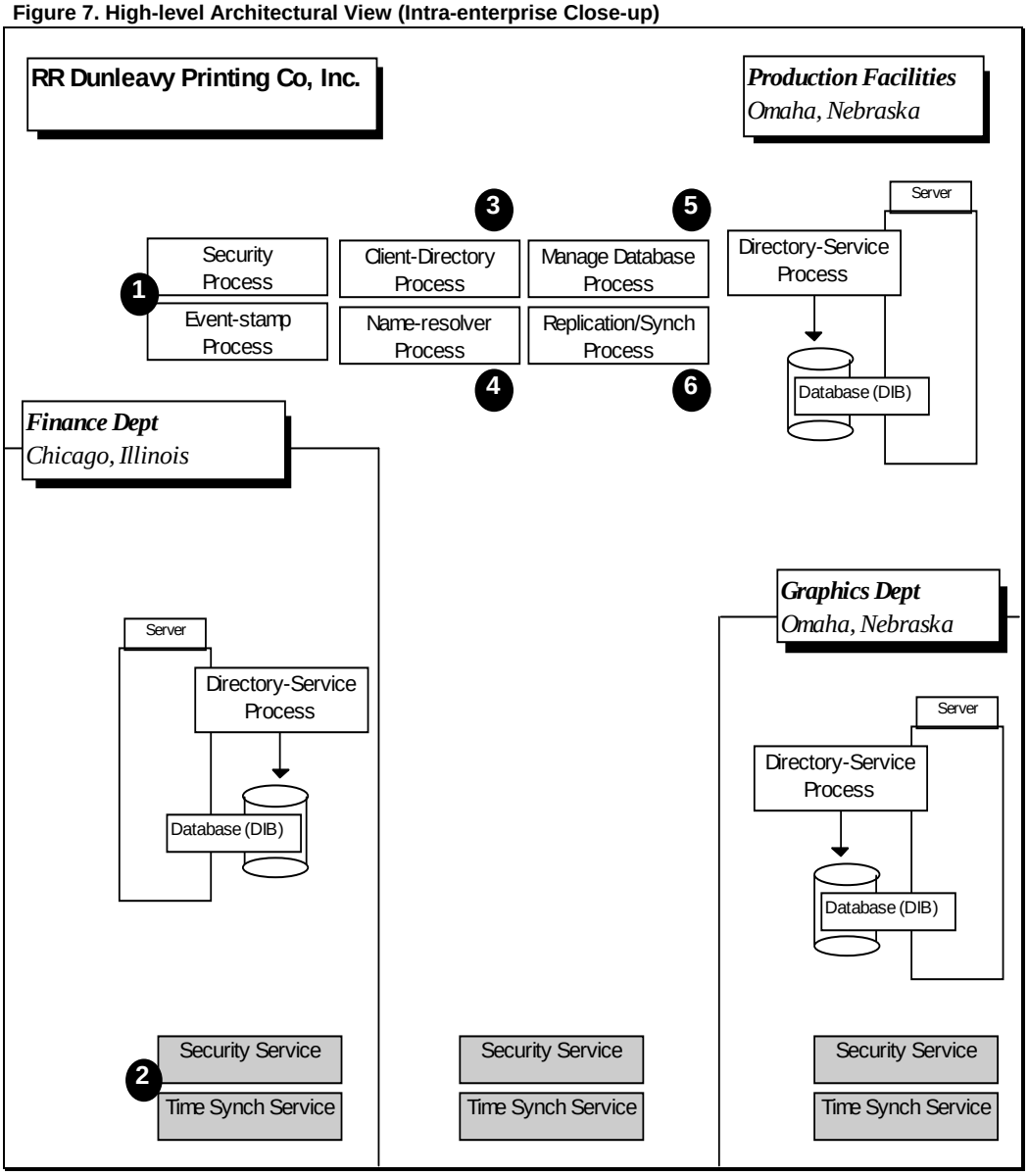
Implementation details surrounding processes associated with directory—security, synchronization, and management—are outside the scope of this paper, but the need of these functions is recognized. Thus, key functions of the directory service process include:

- Security process (which may invoke another service over the network)
- Event-stamp process (which may invoke another service over the network)
- Client-oriented process exposes key functionality to client applications, such as searching functions,
- Name-resolver process is a mechanism that resolves network names and addresses, either sending the information back to the client process immediately or routing the lookup request to another directory service
- Manage-database process includes functions such as adding objects, adding attributes, constructing the attribute and object structure of the database
- Replication/synchronization process works to keep all directories throughout the network in synch with each other. Name-resolver tables are updated as part of this function.

When users want to connect to a file service or print service in a distant department within the company, the directory-service process works behind the scenes to enable them to do just that. Ideally, users won't be thinking about "connecting to a file service" or anything of the sort: they'll simply attempt to open a document, say, that they've been working on from an application's Open dialogue box, and behind the scenes the directory-service process will find the address of the file, which will then open on the desktop. This is the notion of the "unified global view of all resources" in action.

Figure 7 on page 31 provides a more detailed look at the other functions that the directory-service process provides.

Figure 7. High-level Architectural View (Intra-enterprise Close-up)



❶ Before using any network resource, the identity and access-rights associated with the requesting person (or process) must be validated by a security process, with the event recorded in the log.

❷ If these functions are provided by the directory-service process, they are handled by the directory, otherwise a request for these services is passed along to the appropriate services.

❸ The directory-service process accepts requests from client-directory processes (which may or may not be functioning as part of an end-user application) for the contents of the directory database. Requests from a client may include: listing the available contents of the directory database; searching the database; comparing attributes in the database (for a "yellow-pages" type search—"all graphic designers in the marketing department," for example); and find an alias.

❹ The name-resolver process maps names to network addresses. If the name-and-address relationship is available in the local directory database, the name-resolver process returns the address to the client process directly; if not, it forwards the request to another directory-service process independently of location. That is to say that the name-resolver process provides a "directory routing" or "directory broker" mechanism whereby a client process in one enterprise can access the directory-service process in another enterprise, regardless of location. The name-resolver process may also function intra-enterprise as well as inter-enterprise.

❺ The manage database process exposes key management functions to an administration-oriented client interface. These functions include adding entities to the database; deleting entities from the database; modifying entities in the database. In addition, the manage database process enables creating the database structure and defining attribute types.

❻ The replication and synchronization process may be a subset of the manage database process, or it may be implemented as a separate mechanism.

For a more detailed listing of the functions published by a generic directory service, see Appendix C.

In the next section, NAC takes a closer look at the directory information database, its structure and function.

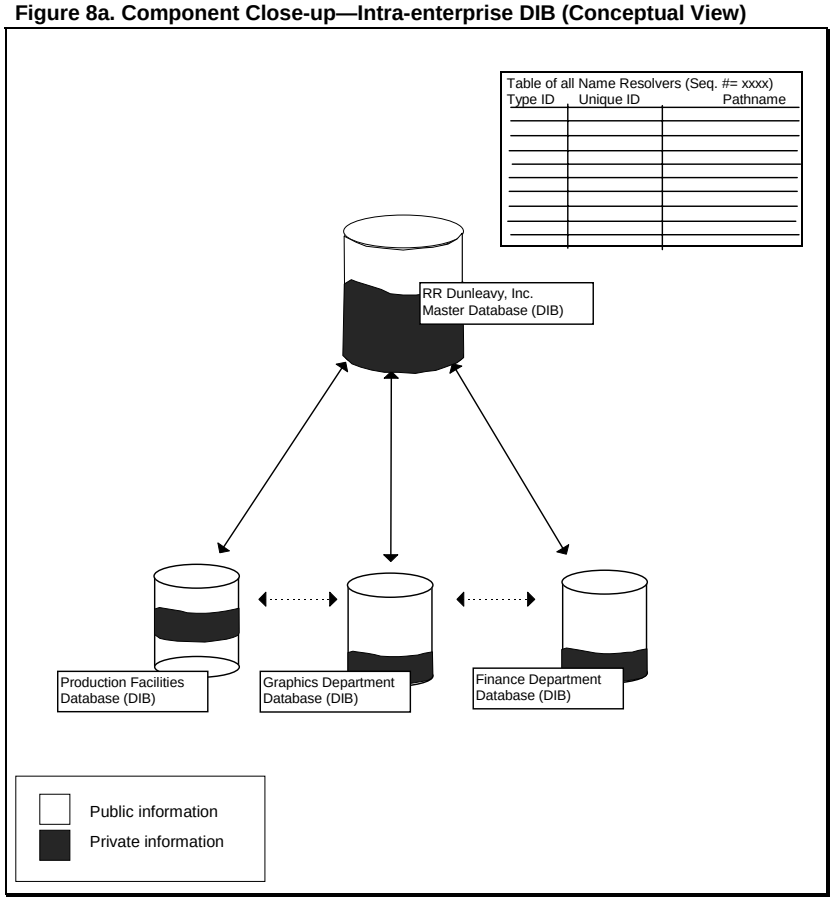
Directory Information Database

Figure 8a on page 33 returns to some themes presented in "*Enterprise Directory Services Architecture*," specifically the hierarchical, distributed nature of the "directory." NAC takes a closer look at the directory information base and its distribution throughout the an enterprise, which should be functionally no different than that which occurs at the global (worldwide) level.

One basic concept depicted in the figure below is that the directory information database is partitioned into multiple databases throughout the organization. The company's directory can be configured to provide a central directory that each department may use to find people and resources (printer, file services, and the like) in other departments. (This can be thought of as a master-satellite configuration.)

Alternatively, however, the company's directory can be configured to allow independent and direct access among departments, without going through the master. The individual department directories may replicate information from the master, information from other directories in other departments. Figure 8a shows both master-satellite and independent directory services.

Figure 8a. Component Close-up—Intra-enterprise DIB (Conceptual View)

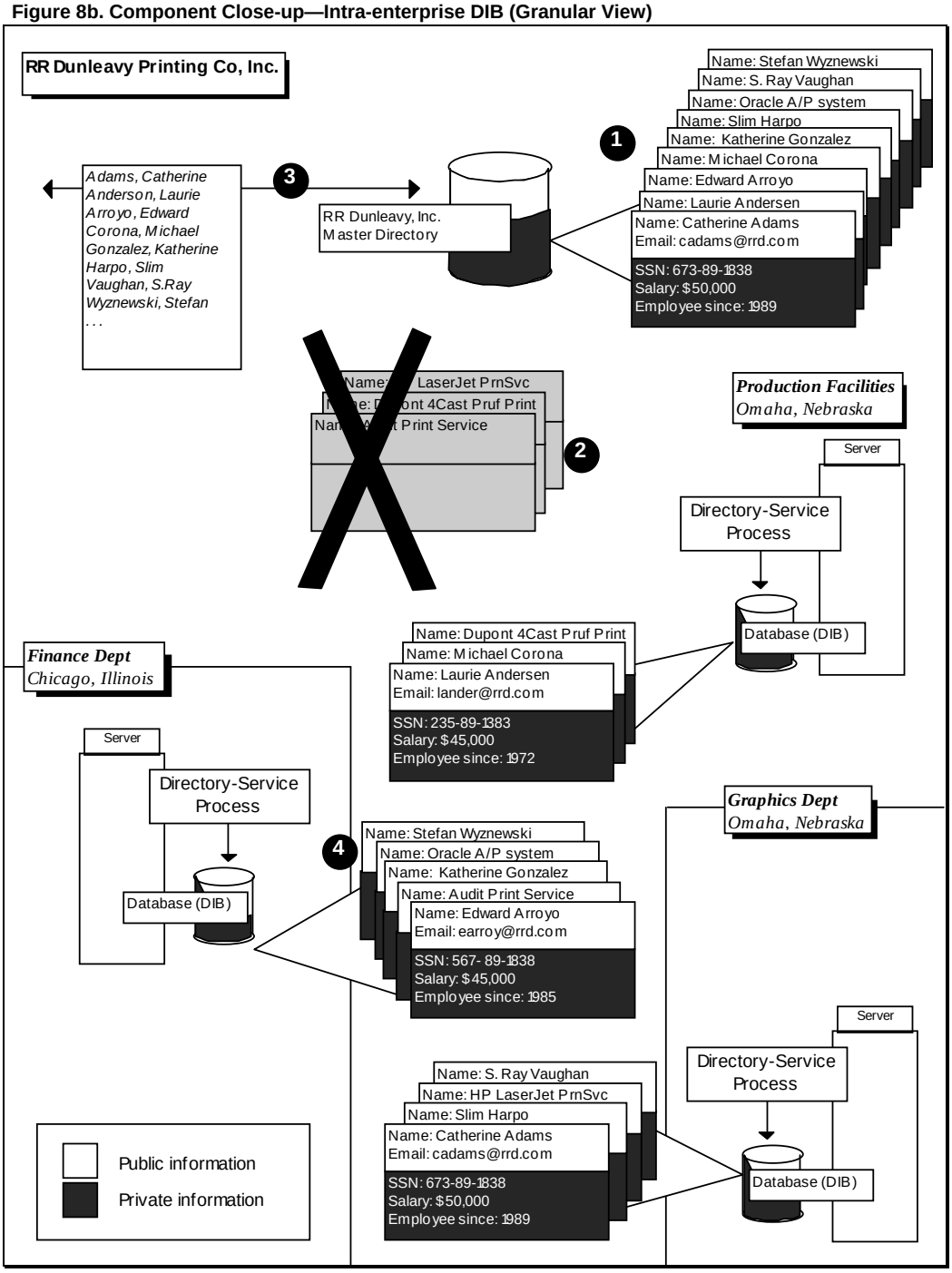


Within an organization, a variety of methods may be combined to provide a robust, distributed directory service.

These methods can be implemented beyond the company, to provide directory information on an inter-enterprise basis as well as intra-enterprise.

Figure 8b on page 34 shows a more granular view of the directory information database and its capabilities as envisioned by NAC. The notion of “public” and “private” information is key, as is the notion of control at the local level.

Figure 8b. Component Close-up—Intra-enterprise DIB (Granular View)



❶ The corporate, enterprise-wide directory DIB (aka, a “master” DIB) contains information consolidated from each of the departmental DIBs. (Note that each “departmental DIB” may in turn be comprised of more than one DIB located on more than one server; these DIBs may contain only a portion of the department’s information.

❷ Note that the master directory does not necessarily contain all elements from the local level. In this example, information about print services, which is only important at

the local level, is not shared with the master directory. On the other hand, information about people and about the information about the corporate-wide payments system (Oracle A/P System) is available on a company-wide basis because everyone in the company can use this information. Such decisions are made by a company when implementing its directory system.

❸ Selected information from the corporate DIB can be shared with the rest of the world by exporting pointers to the information. Note that this is no different than that which occurs intra-company among local directories, or among local directories and a master directory. Also, not all the information held in a company's master directory need be exported; in this example, just the names of individual employees and addresses are published. Again, this is an implementation detail.

❹ Elements in each DIB are managed (changed, added, deleted) at the local level.

Appendix C. Generic Functions of Directory Service Interface Points

Disclaimer: The functions listed below represent only a sample; they are being used to analyze the processes in NAC's generic directory services system. The list is not intended to be interpreted as complete or consistent.

I. Functionality exported by Directory Service

Used by Directory Service

Handshake
- get address types supported
- get name resolve table

Resolve names
- compare sequence #
- compare unique ID
- query local table
- query all tables
- submit to name resolver n+1

Synchronize databases
- compare sequence #
- compare unique ID
- query local table
- request update from master
- submit to name resolver n+1

Directory
- submit request
- begin database entity
- end database entity

Used by client application

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

Session
- open
- close

Access database
- set database context
- get database context
- submit directory access request
- do you accept requests from this id
- do you accept requests to this id

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

Name resolution

resolve name
resolve alias
resolve name list

Attributes

get attribute count
list attributes
get attribute value
compare attribute

Other

get user preferences
set user preferences
register for notification
unregister for notification

USED BY MANAGEMENT SOFTWARE

Database structure and content

create entity
delete entity
rename entity
count entities
enumerate entities
move entities
begin class item
define attribute
define class
get class definition
get class item
get class item count
get syntax count
get syntax definition
get syntax id
list containable classes
modify class definition
put class item
put class name
put syntax name
read class definition
read syntaxes
delete attribute definition
delete class definition
create attribute structure
define attribute
add attribute
delete attribute
create index
begin database entity

end database entity

Database admin

set database entity
get database entity
change database entity attribute

Other

get log data
start service
stop service

II. Functionality exported by Client

Used by Client Process

pass-through lookup request
accept request

Used by Management software

ping
report version
report status

III. Functionality exported by Security service

Used by Directory-service Process

Authentication

login
logout
authenticate requester

Used by Client application

Authentication

login
logout
authenticate requester

Database security

get public key
set public key

IV. FUNCTIONALITY EXPORTED BY EVENT-STAMPING SERVICE

USED BY DIRECTORY-SERVICE PROCESS AND CLIENT

get local time
get GMT
get sequence number
set sequence number

Appendix D. 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.
Attribute	An individual piece of information that describes a particular aspect of an entry in the DIB. For example, "first name," "last name," and "phone number" are all attributes that might belong to a particular entry in the DIB.
Attribute type	A category to which an attribute belongs. For example, "first name" is an attribute belonging to the users in the DIB, but wouldn't be an attribute used for laser printers in the DIB.
Attribute value	The actual contents of the attribute. For example, "first name" might have the value of "John" or "Janet."
B2	A security rating from the US National Computer Security Center that imposes.. Security rating levels include B1, B2, and B3.
C2	A United States government security standard for operating systems which requires that users and applications be authenticated before gaining access to any resources.
COSINE	Corporation for OSI in Europe.
CCITT	Consultative Committee for International Telephony and Telegraphy. The name of an international body that sets telecommunication standards. The name has been changed to ITU-T (International Telecommunication Standardization Bureau—Telecommunication).
DAP	Directory Access Protocol.
DCE	Distributed Computing Environment.
DDE	Dynamic Data Exchange. A feature of Microsoft Windows operating environment and supported applications.
DIB	Directory information base. The complete set of all information held in a directory. The DIB consists of hierarchically related entities.
DIT	Directory information tree. The hierarchical, tree structure, similar to an organization's "org chart," that represents the relationships among all entries held in the DIB.
DME	Distributed Management Environment.

DMI	Desktop Management Interface.
DMTF	Desktop Management Task Force.
DN	Distinguished Name.
DOS	Disk Operating System.
DSA	Directory System Agent.
DSP	Directory System Protocol.
DUA	Directory User Agent.
flat namespace	A namespace where there is no scoping of names.
hierarchical namespace	Each name is defined in the context of a name one level higher.
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.
ISO	International Standards Organization. An independent international body formed to define standards for multivendor network communications.
IT	Information Technology.
ITU-T	The International Telecommunication Standardization Bureau—Telecommunication. Formerly known as the CCITT. Organization that publishes the X-series recommendations as well as international standards in the telecommunications.
Knowledge references	Meta-information (meta-data) about the storage and use of the information in the DIB; information about the information in the DIB. For example, the name of the remote DSA; the address of a remote DSA; the name of the DIT that the remote DSA holds
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.

NADF	North American Directory Forum. The group competing service providers who plan to cooperatively offer a public directory service in North America using the ITU-T's X.500 recommendations.
Namespace	The logical view of the network that is independent of the physical network configuration.
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).
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.
Quipu	A publicly available implementation of X.500 which runs under Unix. (As a sidenote, the name Quipu itself is an American Spanish name that refers to a record-keeping device of the Inca Empire which consisted of a series of variously colored strings attached to a base rope and knotted so as to encode information, used especially for accounting purposes.)
RSA	Rivest, Shamir, and Adelman. The names of the three developers who created the RSA encryption scheme.
Schema	A diagrammatic representation; an outline or a model. Schema is a pattern imposed on complex reality or experience to assist in explaining it, mediate perception, or guide response. In a directory service, a schema defines the directory information tree, its structure and organization.
SIG	Strategic Interest Group. A subset of NAC member companies focused on a particular strategic topic.
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.
VINES	VItural NEtworking System. Network operating system from Banyan Systems, Inc.

WAN	Wide area network.
WOSA	Windows Open Services Architecture. Microsoft's modular framework which relies on a system of snap-in software chunks, called "service provider interfaces," which provide back-end functionality. Microsoft's messaging API MAPI) and Open Database Connectivity (ODBC) are part of WOSA.
X.400	A CCITT and OSI specification for database entity exchange.
X.500	A CCITT and OSI specification for a hierarchical global directory.
XAPIA	X.400 API Association.