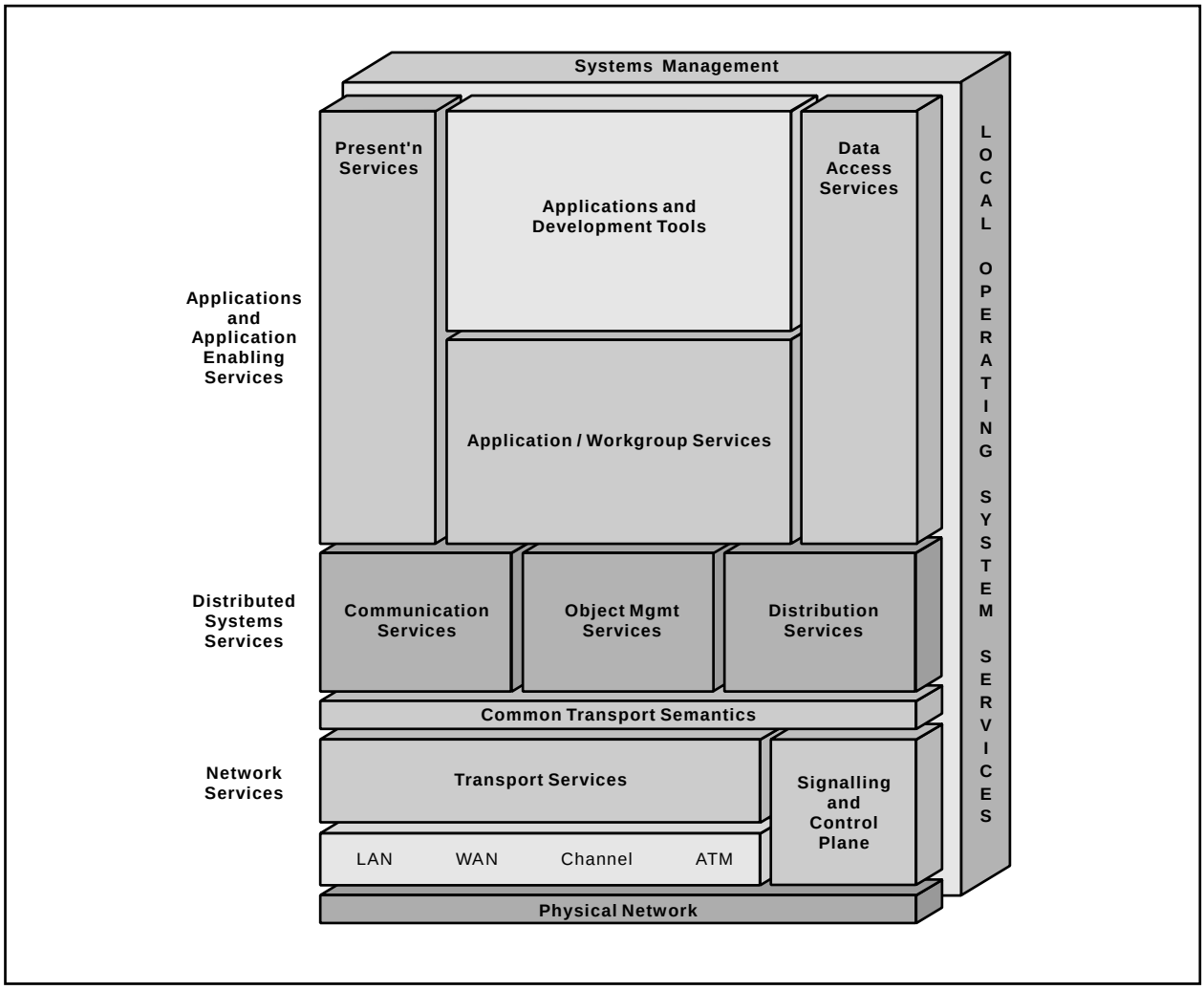
Open Blueprint

Directory Resource Manager



GC23-3915-01

Open Blueprint

Directory Resource Manager

GC23-3915-01

About This Paper

Open, distributed computing of all forms, including client/server and network computing, is the model that is driving the rapid evolution of information technology today. The Open Blueprint structure is IBM's industry-leading architectural framework for distributed computing in a multivendor, heterogeneous environment. This paper describes the Directory resource manager component of the Open Blueprint and its relationships with other Open Blueprint components.

The Open Blueprint structure continues to accommodate advances in technology and incorporate emerging standards and protocols as information technology needs and capabilities evolve. For example, the structure now incorporates digital library, object-oriented and mobile technologies, and support for internet-enabled applications. Thus, this document is a snapshot at a particular point in time. The Open Blueprint structure will continue to evolve as new technologies emerge.

This paper is one in a series of papers available in the *Open Blueprint Technical Reference Library* collection, SBOF-8702 (hardcopy) or SK2T-2478 (CD-ROM). The intent of this technical library is to provide detailed information about each Open Blueprint component. The authors of these papers are the developers and designers directly responsible for the components, so you might observe differences in style, scope, and format between this paper and others.

Readers who are less familiar with a particular component can refer to the referenced materials to gain basic background knowledge not included in the papers. For a general technical overview of the Open Blueprint, see the *Open Blueprint Technical Overview*, GC23-3808.

Who Should Read This Paper

This paper is intended for audiences requiring technical detail about the Directory Resource Manager in the Open Blueprint. These include:

- Customers who are planning technology or architecture investments
- Software vendors who are developing products to interoperate with other products that support the Open Blueprint
- Consultants and service providers who offer integration services to customers

Contents

Figures

Summary of Changes

This paper has been revised to describe:

- Integration of additional directory services to support the Open Blueprint federated namespace. In addition to X.500, Domain Name System (DNS), and Distributed Computing Environment (DCE) Cell Directory Service (CDS), the Lotus Notes Name and Address Book and a Lightweight Directory Access Protocol (LDAP)-based directory service are now included.
- Support of federated names based on Universal Resource Locator (URL) specifications.
- Directory function in Internet and intranet environments based on the LDAP protocol.

Directory Resource Manager Overview

Large scale distributed systems that incorporate many types of network and client/server products can be complex to administer, maintain, and use. Resources are owned and administered by many enterprises and individuals. Access to those resources is often needed by users who are unknown to the resource owner, making it difficult for users to discover, access, and use resources which are available in the network.

In such large, highly decentralized networks, a directory service is the cornerstone of network access. A directory not only lists all published resources, but also provides the information to enable those resources to be accessed by any authorized user. The availability of information such as network addresses, resource manager types and version numbers, or resource gateway information enables dynamic access to the resources from anywhere in the network.

When applications create resources, they publish the resource names. These resources are then known throughout the network by their published names and access to them can be achieved without prior setup or configuration information by specifying the resource name. Resources can then be moved, or access to them changed, without requiring any interaction with the resource users. Furthermore, attributes that provide descriptive information about the resource are also stored in the directory service. These attributes allow for qualitative searching for acceptable resources and resource providers.

What is a Directory Service?

A directory service is a database application that provides an organized way to store, manage, and retrieve information. The directory service can be application-specific or general.

Application-specific directory services generally have a single focus for data content and operations. They can be embedded as part of an application or operating system or associated with an application. Generalized directory services do not have a single focus and present the user with a generic set of operations. These directory services are not associated with or embedded in a particular application. Instead, they are part of an operating system and available for use by any operating system component, service, or application.

The directory service can be accessed using the network or can be as simple as a set of operating system files. The directory service provides access to information by resource name to provide a level of indirection, or location transparency, so that if the resource accessed changes location, the location change is reflected in the directory contents, but the name by which the resource is referenced remains the same. This isolates administrative changes made by the resource provider from the resource user. The directory service is composed of entries and their attributes. An example is a person entry which has attributes such as telephone number and e-mail address.

Types of Directory Services

Directory services are used to solve problems in many areas. Existing directory services have the following capabilities:

- Low-level address lookup services

  Perform fast, efficient name-to-address and address-to-name mappings. An example of this type of directory service is the Domain Name System (DNS).

- Highly volatile directory services

Characterized by update rates that are much higher than access rates. Examples of this type of directory service include network configuration and management information, and distributed object instance location information.

- Classic directory services

  Store information about resources in a computing network. Resources include people, places, and things.

- Meta-directories

  Integrate multiple application-specific and operating system directory services and manage them as a whole. A *meta-directory* stores the data (or subset of data) from other directory services and can contain pointers to information in other directory services it is managing.

- Content based directory services

  Search indexes of information accessible using the Worldwide Web (WWW). Search and index services such as Yahoo and Alta Vista can be viewed as directory services that are built on information contained in the data that the services point to.

Directories versus Databases

The distinction between a directory service and a database service is beginning to blur. Directory services store data and hence act as a database of sorts. On the other hand, some databases are really used to hold directory type information. The distinction is further blurred by the approach many directory service implementations are taking in making use of database services for storing large amounts of information.

The biggest differences between directory services and database services revolve around areas of replication and transactional integrity. While directory services have advanced the technology in terms of replication of directory information to support high availability and concurrent access, database services have advanced the technology in terms of transactional integrity and referential integrity for processing concurrent requests. Traditionally, directory services have been characterized as holding relatively static information which must be available for access. The design of directory services has generally accepted that there can be some latency between an update made to data in the directory and the changed data being available on all replicated sources and accessing applications.

The fact remains that while a database or a file system can provide implementation support for a directory service, directory services have a unique presence in the industry, and are key infrastructure services. The Open Blueprint Directory resource manager provides this required infrastructure service.

Today's Problems and Current Solutions

A common problem facing most computing organizations today is the myriad of directory services that exist and must be managed across the enterprise. Almost every application uses a directory service of some sort to store important information, usually tailored to the application and hidden from the end-user. Configuration files, IP addresses, e-mail addresses, mail alias files, distributed object locating services, network operating system directories, and workstation registries must be managed. Data is duplicated among many of these directory services, usually in slightly different formats. Immense administrative effort is required to attempt to keep all of these entries synchronized.

In the past few years, a number of directory services have begun to stand out as directory services in their own right, not just as hidden services used by the application with which they were supplied. Among these by type are:

- Network Operating System (NOS) (Novell NDS, Banyan StreetTalk)

- Name Services (Microsoft Exchange, Novell Netware Bindary, Distributed Computing Environment (DCE) Registry and Cell Directory Services (CDS), SunSoft NIS+, DNS
- General Purpose (X.500, Netscape Directory Server)

Most directory services today exist in the enterprise. As the Internet community evolves its directory services and as the use of directory services becomes more widespread, the distinction between enterprise and global or Internet directory services will blur. Data in all of these directory services will need to be seamlessly accessed, subject to authentication and access controls. It will be important for Internet and enterprise directory services to fit together to form a cohesive namespace that appears as a single directory service.

Open Blueprint Directory Resource Manager

The Open Blueprint Directory resource manager solves today's problems by federating these communicating and interoperating directory services. The thought that a single directory service could be employed to organize all directory information is a very attractive idea, but not practical. Due to the proliferation of directory services and the requirement to remain compatible, a single directory service handling all client requests becomes an insurmountable task. A better approach is to look to well-established open standards for a common ground on which these individual directory services can communicate and interoperate.

The Open Blueprint Directory resource manager and associated naming formats are based on the evolving Internet directory services/X.500 and Lightweight Directory Access Protocol (LDAP), X/Open DCE Directory Services, Object Management Group (OMG) Naming Services, and X/Open Federated Naming. Although the LDAP acronym refers to a protocol, LDAP encompasses several items: an Application Programming Interface (API), a protocol, and a Uniform Resource Locator (URL). LDAP is a simplified subset of X.500.

Definitions and Model Overview

X/Open Federated Naming provides the following definitions (the definitions are excepted from X/Open's *Federated Naming: The XFN Specification*):

- *A composite name* is a name that spans multiple naming systems (directory services). It consists of an ordered list of zero or more components. Each component is a string name from the namespace of a single naming system. (This is also known as a *federated name*).
- A *federated naming system* is an aggregation of autonomous naming systems that cooperate to support name resolution of composite names through a standard interface. Each member of a federation has autonomy in its choice of operations other than name resolution.
- *A federated naming service* is the service offered by a federating naming system.
- *A federated namespace* is the set of all possible names generated according to the policies that govern the relationships among member naming systems and their respective namespaces. (A federated namespace is the logical union of one or more namespaces.)
- In a federated naming system, a *naming system boundary* is the point where the namespace under the control of one member of the federation ends, and where the namespace under the control of the next member of the federation begins. This is sometimes referred to as a *junction*.

The Open Blueprint Directory resource manager supports a federated namespace. Each namespace can be implemented by a different directory service. Each directory service, supporting its partition of the federated namespace, has its own characteristics such as replication, caching, distribution, a security model for authentication and authorization, and attribute support. A directory service can provide a generalized service or resource-manager-specific services. A federated namespace allows the data in a particular partition of the namespace, backed by its specific directory service implementation, to remain in that place. This allows new directory services to be added and existing directory services to be replaced or deleted.

The Open Blueprint Directory resource manager (the federation) is a distributed, replicated "database" service. It is *distributed* because the information that forms the database is stored in different places; information about one group of users and resources might be stored in one directory server, while information about a second group of users and resources is stored in a different directory server. The Open Blueprint Directory is *replicated* because information about a name or group of names can be stored in more than one location for higher availability.

The Open Blueprint Directory resource manager namespace (the federated namespace) consists of a hierarchical set of names that have associated attributes. Given a name, its associated attributes can be looked up in the Open Blueprint Directory service. Given an attribute or set of attributes, a name or set of names satisfying those attributes is returned. For example, given the name of a print server, the Open Blueprint Directory service can return the printer's location. Or given a print location, the names of print servers for that location can be returned. The Open Blueprint Directory resource manager gives distributed system users a well-known, central place to store information, which can then be retrieved from anywhere in the distributed system.

Namespace

The federated namespace is the set of names used by the Open Blueprint Directory resource manager. It is based on the federation concept. It is hierarchical. It consists of two classes: global and enterprise. The relationship of these two classes of namespaces (and hence naming systems) is shown in Figure 1 below. The hierarchical depth of these two classes is not limited to that shown in Figure 1. It shows the combination hierarchy. These namespaces are logically connected (or federated) together using what can be called junctions. A *junction* is an entry in the namespace that contains binding information to enable communications between different namespace partitions of the Open Blueprint Directory resource manager.

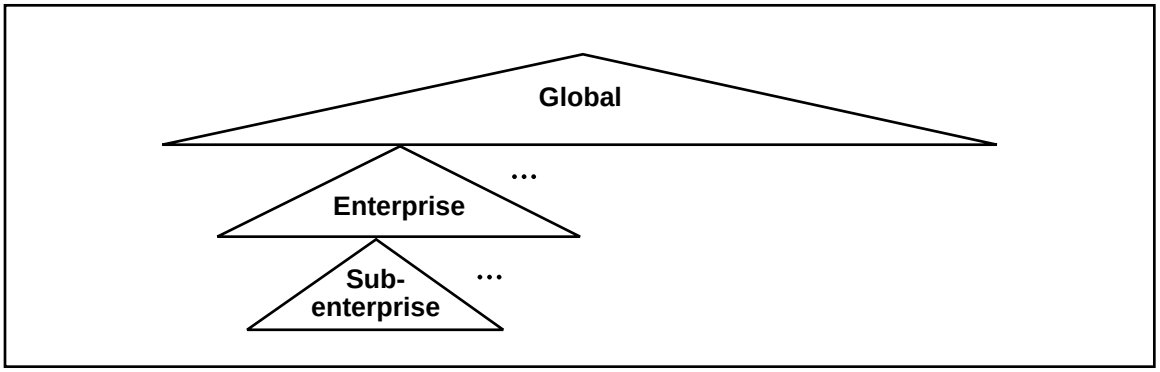


Figure 1. Global Namespace

The global namespace has a world-wide scope. The types of entities named at this global level include countries, states, companies, government, universities. X.500 Directory Services and the Internet DNS are examples of global naming systems. Resources that are advertised as global are named in this space.

Enterprise Namespace

Enterprise namespaces are located beneath the global namespace. The enterprise namespace defines one or more scopes of administration. The enterprise namespace can be single or multiple namespaces, and these namespaces can be junctioned. These junctioned namespaces can provide specific services such as the registry (users), file, and print services.

What is Named?

Any resource can be named: people, places, things. Examples of resources are files, printers, services, users, and groups. Resource information (names and attributes) is placed in the namespace. This information consists of data necessary to locate and connect to the resource, and descriptive information useful in searching for a resource.

Because the namespace is hierarchical, a resource's name reflects this hierarchy. The full name of the resource consists of the complete set of names from the top of the namespace down to the resource's specific name.

Names

A name is how an application refers to a resource in the namespace. A string syntax allows the application to express the name in a human-readable form. Each naming system has its own defined string syntax. Because the federated namespace is composed of multiple namespaces each representing a possibly different naming system, a name can cross namespace boundaries. A naming syntax is required to create a federated namespace and still support the system-specific naming syntaxes that will continue to exist.

Federated names can cross namespace boundaries. A federated name contains zero or more junctions. This approach brings existing and new resources that may reside in disparate namespaces into the federated namespace. There is no need for data residing in one namespace to be moved to another namespace for access.

Given multiple junctions in the namespace, multiple federated name specifications can be used within a resource name. Federated naming defines a common global root for all resource identification. Without this common global root, specific knowledge of how to find the root of each naming system is needed to locate resources. There are two forms of names:

- Universal Resource Locators (URLs)

  Composition of URLs is defined by Internet Engineering Task Force (IETF) Request for Comment (RFC) 1738 *Uniform Resource Locator (URL)* (and others). The general form of a URL is:

  `<scheme>://<user>:<password>@<host>:<port>/<path>`

  where each scheme tailors its scheme-specific part (the part after the //). The LDAP and HyperText Transfer Protocol (HTTP) URLs are examples of schemes related to directory information. The portions of the URL prior to the `<path>` can be viewed as describing the global root of the namespace.

  When the `<scheme>` and connection point (`<user>:<password>@<host>:<port>`) are determined, the path portion of the URL is scheme specific. Schemes such as LDAP can federate different naming systems by mapping an LDAP URL format path to the naming-system-specific string syntax. In this way, a federated namespace can be provided that eliminates differences between naming system string syntaxes. The user sees a uniform namespace.

- Federated Names

  Composition of federated names is defined by the Internet Engineering Task Force (IETF) RFC *A String Definition of Distinguished Names* and X/Open's Federated Naming.

  – IETF LDAP distinguished names

    LDAP names have a uniform syntax. Namespace boundaries are not apparent. A component of a name is called a *relative distinguished name* (RDN). An RDN represents a point within the namespace hierarchy. RDNs are separated and concatenated by using the comma (,). Each RDN is typed. RDNs have the form `<type=value>` for single valued RDNs. The plus sign (+) is used to form multi-valued RDNs, for example, `<type=value>+<type=value>`. A concatenated series of RDNs is known as a *distinguished name* (DN). The DN is used to represent an object and the path to the object in the hierarchical namespace. A URL format for LDAP has been defined that includes a DN as a component of the URL.

  – X/Open federated names

    X/Open federated names can cross naming system boundaries. Names that cross naming system boundaries are concatenated by the forward slash (/). The name syntax between forward slashes is specific to and defined by the naming system for that part of the name. Name components in X/Open federated naming can be typed or untyped, reflecting the different name formats supported by naming systems. Name components that are typed consist of a type and a value

separated by the equal sign (=). A X.500 name such as /C=US/O=ABCcompany is an example of a typed name. An untyped name consists only of values such as abc.com; DNS and the DCE Cell Directory Service (CDS) are examples of naming systems that use this format. Because the name of an object can be a composite name, its X/Open federated name can have both typed parts and untyped parts.

Figure 2 provides examples of URLs and federated names to show two different forms of federated names that contain equivalent information.

Figure 2. Name Examples

Things to Name	**URL**	**Federated Names**
Web page	http://<DNShostname>/<path>	<path>
Distributed file services file	dfs://<cell>/<path>	/.../<cell>/<path>
NFS served file	nfs://<remotehost>/<path>	/<localmount>/<path>
LDAP directory entry	ldap://<hostname>/cn=John Q. Public, ou=Marketing, o=IBM, c=US	c=US, o=IBM, ou=Marketing, cn=John Q. Public
X.500 directory entry	ldap://<hostname>/cn=John Q. Public, ou=Marketing, o=IBM, c=US	/.../c=US/o=IBM/ou=Marketing/ cn=John Q. Public
DCE CDS entry	ldap://<cell>/<path>	/.../<cell>/<path>
DCE Registry entry	ldap://<hostname>/principal=johnq, ou=cellname, o=IBM, c=US	/.../<cell>/sec/principal/johnq

Contextual Names

For ease of use and programming, contextual names are supported. A *contextual name* is a name that represents some portion of a composite (federated) name. Contextual names are valid only in a context in which the remainder of the composite name has been provided.

How context is established is currently an implementation choice of each resource manager. For existing resource managers, contextual names are often used to map their current programming interfaces to the federated namespace. For example, some portion of a resource manager-specific name (such as a device name, disk name, or group name) can be treated as a symbolic reference to an environment variable that contains the remainder of the composite name.

The URL format described in IETF RFC 1738: *"Uniform Resource Locators (URL)"*specifies that contextual URLs only provide a portion of the <path> specification. This portion is interpreted relative to the current URL, which is used as the context.

With X/Open Federated Naming, each point in the federated name can be specified as a context from which relative names can be specified. A contextual name does not contain the /... prefix.

Aliases and Softlinks

An alias is an alternate name. It can be a composite (federated) name or a contextual name. A DCE CDS softlink is a form of alias; it is a name that contains a pointer to the real name. Alias entries in an X.500 directory service are another form of alias. When a name that is referenced is an alias, the entry returned is the directory entry that the alias points to and not the entry itself.

Structure and Characteristics of a Directory Service

Figure 3 depicts the general functional composition of a directory service. A directory service supports a naming syntax and namespace and is composed of several parts:

- Client
- Server
- Information model
- Schema
- Directory information tree (DIT) and directory information base (DIB)

Specific directory services are described in “Directory Services Descriptions” on page 19.

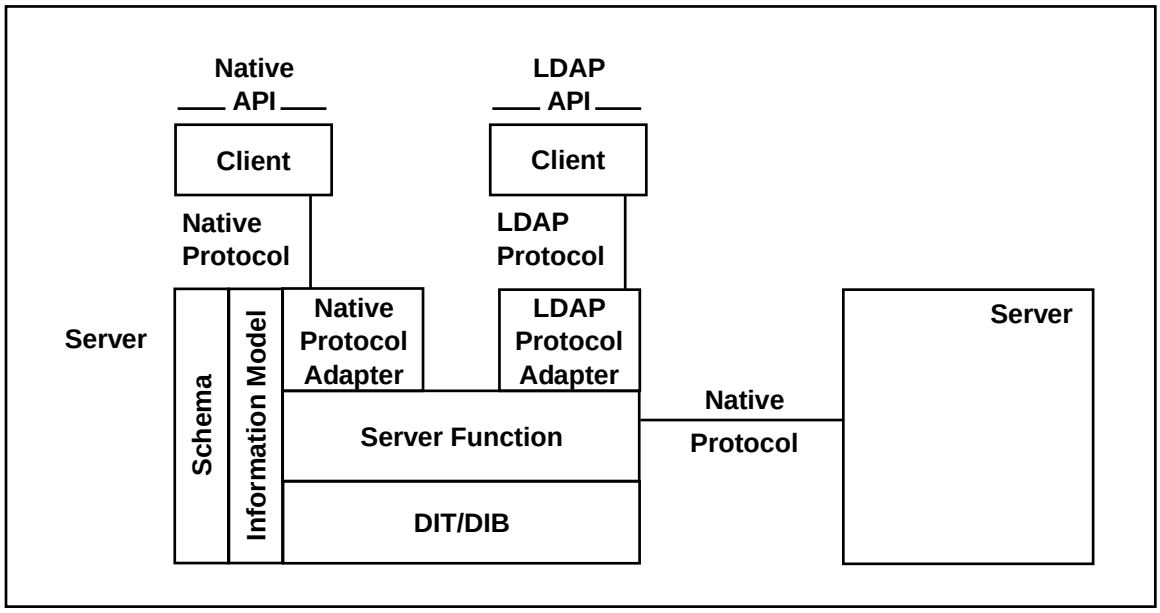


Figure 3. Composition of a Directory Service

Client

The client function implements a user or programming interface to access information stored in the directory service. This interface emits a protocol to communicate with a directory server. Each directory service generally has its own native API for programming and native protocols for network access. This makes client communication to several different directories difficult.

LDAP provides a directory neutral protocol for accessing information in a directory service. The LDAP API (both procedural and object-oriented Java) provides a programming interface independent of any specific directory service and emits the LDAP protocol. The programming interfaces provide a comprehensive set of operations: read, list, search, write/modify, and control. Directory servers accept the LDAP protocol in addition to the native, directory-service-specific protocol.

This functional structure continues to support the native APIs and native protocols for the Open Blueprint directory services; examples are XDS/XOM and NSI client APIs access to DCE CDS information through the CDS protocol in the DCE directory service. The addition of LDAP access to directory servers enables access to multiple directory services through a single protocol, the LDAP protocol, and a single client API, the LDAP API.

Server

The server accepts the client access protocol, performs server functions such as accessing information in the data store, sending referrals (redirecting the client to another server that can satisfy the request), chaining the request to another server on behalf of the client, and replicating data between servers. There can be a separate protocol for server-to-server communications.

Partitioning: The server can support partitioning (distribution). Directory partitions are logical divisions of the database that can be placed on different physical media. This partitioning of information is transparent to users, so the network looks like a single collection of resources. This distribution of data allows the system to scale, increases performance, and makes management easier.

Replication: The server can support replication. Replicas are created to improve performance and availability of the data. If a server is down and a replica of the data at that server is available on another server, the user request can still be satisfied. Replicas can be of various types:

- Single read/write master with multiple read-only copies where updates are made to the master and propagated to the read-only copies
- Multiple read/write copies where updates can be made to any copy and are propagated to the other copies

Replication can occur in near real-time when updates are made or at scheduled update intervals.

Name Resolution: Name resolution is handled through both client and server processing. In a federated system, as a directory service boundary is crossed, the initial server contacted might not be able to satisfy the request but might know what other server to contact. Two methods are used to complete name resolution:

- Referral, where the server informs the client of the next server to contact to continue resolution
- Chaining, where the request is passed from the current server to the next server on behalf of the client until the request is resolved

Actual resolution can involve both methods.

Information Model

Characteristics of the information model are:

- The information is generally organized in a hierarchical structure
- The operational behavior of the system is dominated by queries versus updates
- Directory information changes infrequently

Each point in the hierarchy is either a leaf or non-leaf entry in the hierarchy. Non-leaf entries can serve as context for contextual name resolution. Both leaf and non-leaf entries can store information. Information is typically stored in attribute value pairs. Attributes are named by either string format names, object identifiers (OIDs) which have string format names associated with them, or universal unique identifiers (UUIDs), which also have string format names associated with them. The directory service information model can support both single and multi-valued attributes. In the case of multi-valued attributes, there is no order to the attribute values. An entry, then, consists of a set of attribute value pairs where one or more values can be stored for an attribute.

Directory Schema

The directory schema is a set of rules that define the structure of the directory hierarchy as well as the set of attributes that can be stored in entries in the directory service. The schema defines the specific types of information held in the database, and defines the syntax of the information. Each directory service defines its own schema. Some schemas are standards-defined, such as those defined by the X.500, X.400, and IETF specifications.

Directory Information Tree and Directory Information Base

The directory service stores information as entries (and their attributes) about resources. The entries are organized into a directory information tree (DIT) that is generally hierarchical. The directory information base is the collection of entries that make up the DIT.

Open Blueprint Directory Resource Manager

The Open Blueprint Directory resource manager is composed of many different directory services and their corresponding namespaces. These namespaces are federated via IETF LDAP distinguished names. DNS and X.500 serve as the global namespaces under which enterprise namespaces are federated. The services in Figure 4 that are shaded are not supplied as part of the Open Blueprint Directory resource manager but are included to show how they fit in the federation.

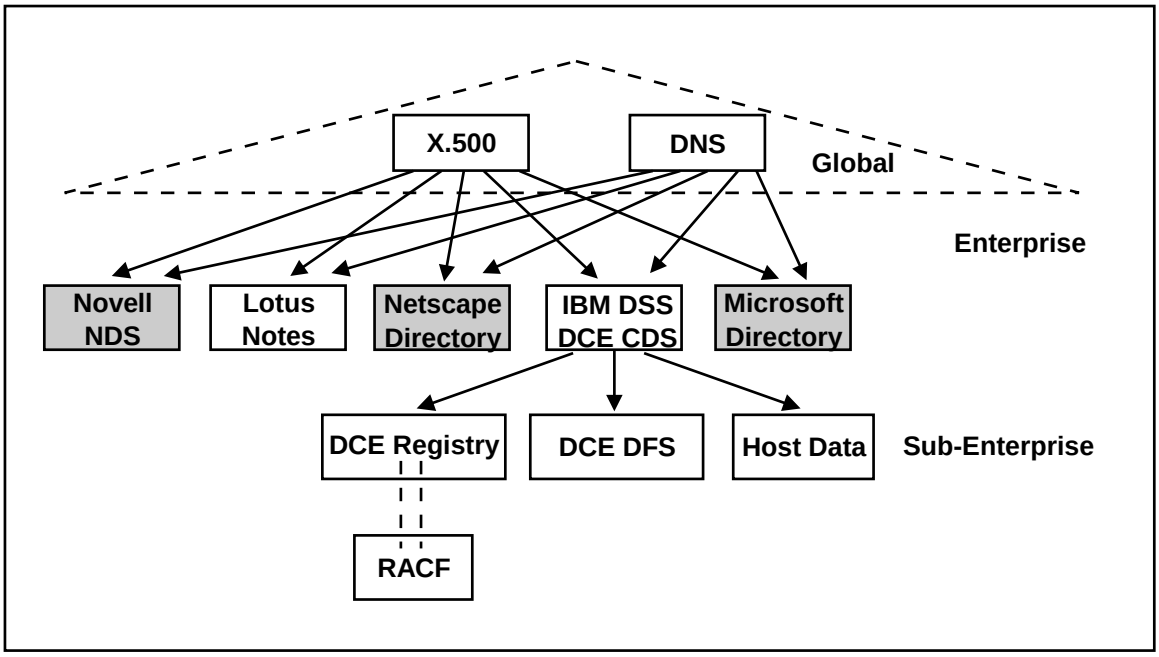


Figure 4. Federation of Open Blueprint Namespaces

Types of Services

The Open Blueprint Directory resource manager provides the following types of directory services:

- Global Directory Services
  - DNS

    DNS is a distributed, replicated database. Copies of information can be stored in multiple DNS servers, and the information can also be cached. DNS is generally used to map between Internet host names and Internet addresses. DNS has a defined set of interfaces for access to the data it keeps. A protocol is defined for communication between the DNS client and server. TCP/IP applications depend on this service for host name-to-address resolution. DNS servers now allow dynamic updates to be made to the DNS namespace. Dynamic updates allow dynamic IP address assignments to be accessible from the DNS server.

  - X.500

    X.500 is a more general directory service that supports names, bindings, attributes, and search. X.500 has defined protocols for client/server and server/server communication. It is a distributed, replicated directory service that is based on the *ITU International Telecommunications Union-T Recommendation X.500* international standard. Copies of information can be stored in multiple X.500 servers, and the information can also be cached.

- Enterprise Directory Services
  - IBM Directory and Security Server (IBM DSS)

    These services are DCE services. DCE Cell Directory Services (CDS) allow DCE servers to publish their existence and DCE clients to locate DCE servers. DCE Remote Procedure Call (RPC) applications that use the DCE RPC NSI routines to export and import server bindings use the Cell Directory Service.

    The DCE Registry contains user and group definitions for an enterprise as well as the DCE environment. In addition to DCE-based private key authentication information, extended registry attributes allow other user specific information to be stored in the DCE Registry. A user can have a single identity across an enterprise.

    The IBM DSS directory service is a superset of the DCE Cell Directory Services and DCE Registry and is used by resource managers to store non-RPC and non-DCE information. As such, it forms an enterprise directory service.

    The RACF/MVS Registry can be linked with the DCE Registry. Because the RACF/MVS Registry can contain a large number of users for an enterprise, it can also be viewed as an enterprise directory service. Both users and resources can be defined to RACF/MVS.

  - Lotus Notes Name and Address Book

    The Lotus Notes Name and Address Book defines users, stores mail address information, and stores server information for collaborative environments. The Name and Address Book can serve as an enterprise directory service. The Name and Address Book can be extended through Lotus Notes database forms to store information about other resources in the enterprise.

Lightweight Directory Access Protocol (LDAP)

IBM's Directory and Security Server (DCE CDS and DCE Registry) and X.500 are accessible through the LDAP protocol. LDAP access makes information stored in DCE directories available to Internet browsers and applications and allows LDAP to be used to establish communication between DCE cells.

Lotus Notes and cc:Mail are accessible through the LDAP protocol. LDAP access makes information in the Notes Name and Address Book and cc:Mail files available to Internet browsers and applications.

Meta-Directory

Most organizations, large and small, oversee a variety of heterogeneous directory systems, primarily associated with messaging/groupware applications and network operating systems. The technical and political legacy resulting from the growth of network computing has resulted in a proliferation of directory systems that are proprietary and do not generally accommodate one another.

As the technical and administrative headaches of many large enterprises increase, so does the urgency for directory integration and the need to manage multiple namespaces from a single point of overall control. This integration needs to occur both from the management and the end user perspectives as the growth of directories continues and usage accelerates.

Directory services provide a simple way of naming, finding, accessing and protecting resources over both space and time. Meta-directory services provide a universal way of naming, finding, accessing and protecting resources not only over space and time, but also across system boundaries. A meta-directory service is designed specifically to integrate multiple directories that are connected to it and represents the union of attributes for meta-directory objects from all directories in an organization.

The meta-directory accommodates the selective bi-directional propagation of objects and object-attributes to and from any connected directory. From a management perspective, the meta-directory's primary role is the creation and maintenance of the relationships between objects and attributes across numerous directory namespaces. As a result, the meta-directory supports both the meta-directory namespace and the namespace associated with each of those connected directories.

Integration of Enterprise Directory Services

There is, and will continue to be, a multiplicity of directory services used in the Internet and enterprise. These directory services are federated to provide the Open Blueprint Directory resource manager. This federation has a namespace which all the constituent directory services support. The LDAP protocol is used to form this federation (see Figure 5). Because enterprise directory services support the LDAP protocol, and directory services use referral and chaining mechanisms to access other directory services that support the LDAP protocol, a namespace is formed that federates the constituent directory services.

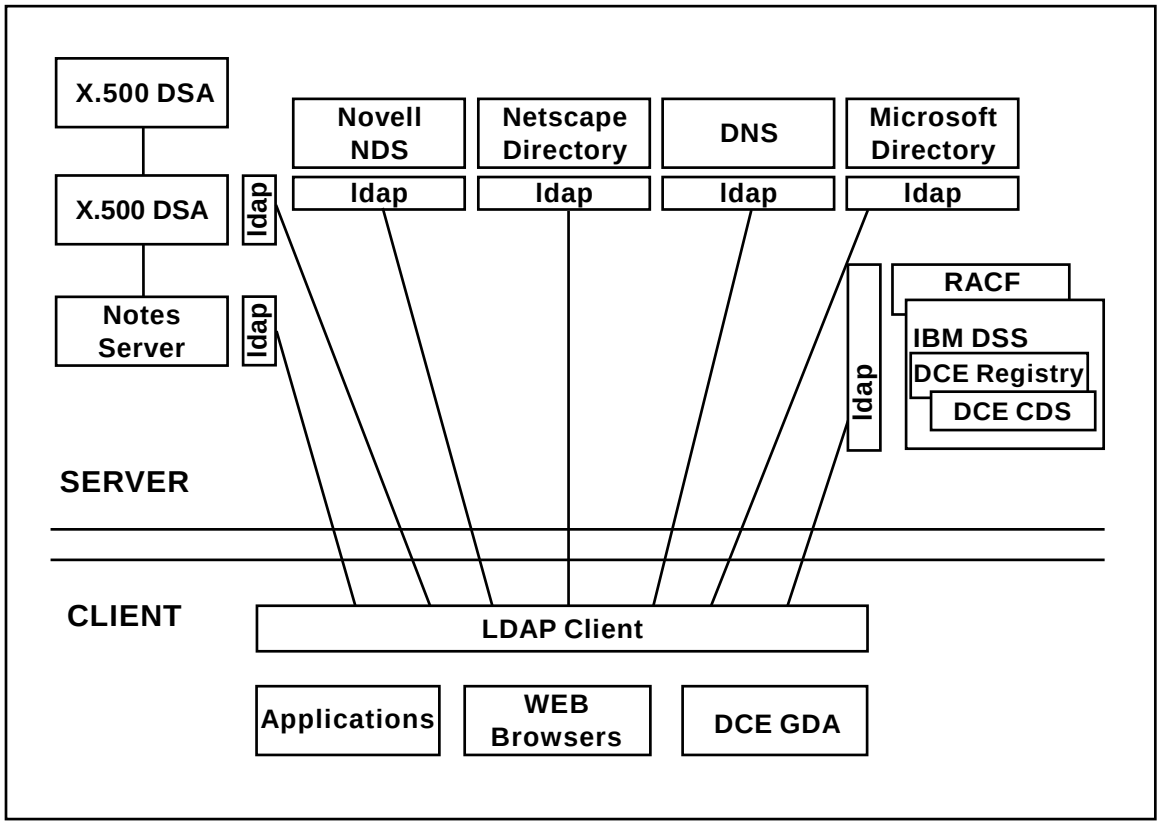


Figure 5. Directory Federation with LDAP Protocol

The IBM Directory and Security Server (IBM DSS) is federated in the LDAP namespace. LDAP access to the DCE Registry and CDS allows user and resource information stored in DCE to be brought into the federation. LDAP access to RACF registry information allows RACF stored user and resource information to be brought into the LDAP based directory service federation. Because the DCE Global Directory Agent (GDA) uses the LDAP protocol for locating other DCE cells, DCE cells are federated into the LDAP namespace.

LDAP access to the Lotus Notes Name and Address Book allows information stored in the Name and Address Book to appear as a partition of the directory service federation.

Novell's NDS and Microsoft's directory are shown in Figure 5 for completeness. The information shown is derived from Novell and Microsoft press releases.

Directory Services Descriptions

The generalized directory services that are federated using the LDAP protocol are described in this section.

X.500

Names

An X.500 name is a typed name. A typed name consists of an ordered set of <type=value> pairs. Each <type=value> pair is called an *attribute value assertion* (AVA). One or more unordered AVAs together form a *relative distinguished name* (RDN). An X.500 name is an ordered set of RDNs. Examples of distinguished names are:

```
c=US/o=IBM

c=US/o=IBM/ou=Austin,dept=UVZS/cn=John Q. Public
```

In the second example,

```
ou=Austin,dept=UVZS
```

is an RDN that contains two AVAs.

Each distinguished name represents an entry in the X.500 Namespace.

In the Open Blueprint directory federated namespace, X.500 names can be expressed as LDAP distinguished names (DN). The DN is part of the LDAP URL. An example of this is:

X.500 style: `c=US/o=IBM/ou=Austin,dept=UVZS/cn=John Q. Public`

expressed as:

LDAP DN: `cn=John Q. Public,ou=Austin+dept=UVZS,o=IBM,c=US`

LDAP URL: `ldap://ibm.com/cn=John Q. Public,ou=Austin+dept=UVZS, o=IBM,c=US`

Namespace

The X.500 namespace is hierarchical. Each entry contains attributes. Attributes are typed, and multiple values can be stored for each attribute. Multiple values have no order. Each attribute type conforms to an attribute syntax. Examples of syntaxes include PrintableString and Integer. ASN.1, a standard type declarative language, is used to describe attributes stored in the X.500 namespace.

Special entries exist in an X.500 namespace called aliases. Aliases represent links to other partitions of the X.500 namespace. When the distinguished name of an X.500 alias is used, the entry accessed is the entry to which the alias refers. An example subset of the namespace is shown in Figure 6 on page 20.

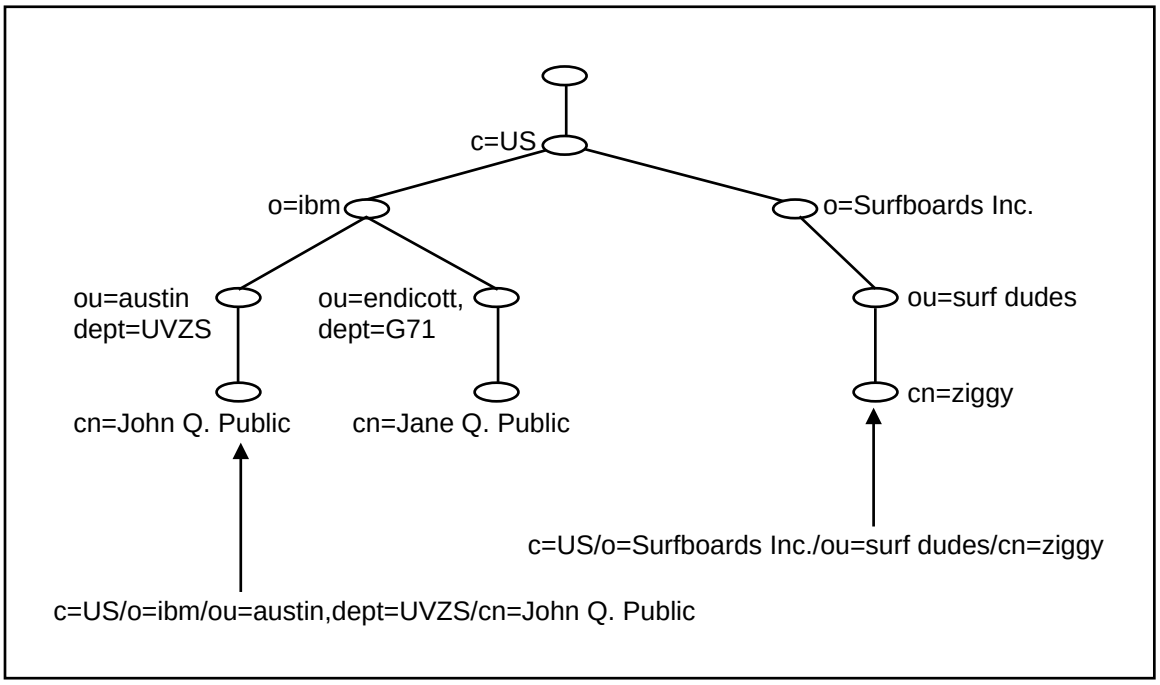


Figure 6. X.500 Namespace

Structure

The general structure of a X.500 Directory Service is shown in Figure 7.

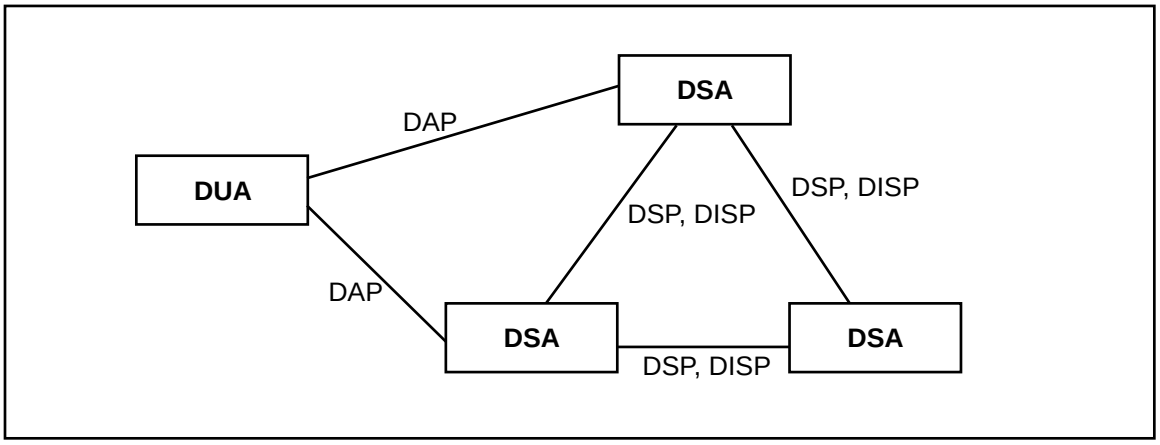


Figure 7. General Structure of a X.500 Directory Service

Client: Access to information stored in a X.500 directory service is accessible only through a directory user agent (DUA). The DUA communicates with one or more directory system agents (DSAs) to locate and extract the information from the directory information database (DIB). DUAs communicate with a DSA using the directory access protocol (DAP).

A DUA can have a number of different interfaces, including graphical-user-interface-based programs and C language callable programming interfaces. One of the more common interfaces is the X/Open Directory Services / X/Open Object Management (XDS/XOM) programming interface. This standard interface (usually as a set of C language calls) provides methods for accessing and modifying information stored by the X.500 directory service.

Server: The directory server in an X.500 directory service is the DSA. The DSAs accept requests from DUAs and other DSAs. DSAs communicate with each other using the directory service protocol (DSP) and directory information shadowing protocol (DISP).

Partitioning: To allow for concurrent access, load balancing, and availability, each DSA stores and manages a partition of the X.500 namespace. These partitions represent sub-trees in the X.500 namespace. Sub-trees can be defined under sub-trees and as a consequence, the DSAs can form a hierarchy.

Replication: Replication is used to maintain more than one copy of information in the DIB. This is usually done for availability purposes. A DSA can replicate information it stores to other DSAs. DSAs replicate information using the DISP. A master/copy approach is used in which the information for an entry stored at one DSA is deemed the master version of the information. If an update is made to the information, all other copies of the information are modified at a later time. Updates to copies of information can be immediate or periodic.

Name Resolution: Two basic methods are used for name resolution in X.500: referral and chaining. With referral, if the DSA that a DUA contacted cannot resolve the name, the unresolved portion of the name is returned to the DUA along with information about another DSA that can be contacted to further resolve the name. With chaining, the DSA will contact another DSA on the DUA's (and thus the user's) behalf, returning a result to the DUA only after the chained request is complete.

Information Model: The information model for X.500 directory information can be modified and extended. Information is stored in an X.500 directory into entries that contain attributes. Attributes are (possibly multi-valued) `<type=value>` pairs in which the type is defined by an object identifier (OID) and the value has a defined syntax. An entry can contain multiple attributes. Entries are organized hierarchically by their distinguished name. Entries whose distinguished name contains the distinguished name of another entry as a prefix are considered to reside "under" the latter entry in the hierarchy.

Schema: An X.500 schema defines rules for distinguished names and what attributes an entry can contain. A standard X.500 schema exists and is recommended, although its use is not required.

Though, in general, a distinguished name is defined as an ordered set of RDNs, a directory schema is employed to force the distinguished names to conform to a set of rules. Typically these rules take the form of "the top of the X.500 namespace will contain entries named using the country name (C) attribute type" or "all entries immediately below the country name entry must be organization (O) entries". This enforces some structure to the upper layers of the X.500 name hierarchy.

To organize the information stored in X.500 directory entries, the schema defines object classes. An object class consists of a set of mandatory and optional attributes. Any entry in the X.500 directory has an object class associated with it. Thus, each entry in the X.500 directory contains a set of mandatory attributes and (possibly) a subset of the optional attributes, based on the entry's object class and the defined schema.

Directory Information Tree and Directory Information Base: The method of storage for the DIT of an X.500 directory is implementation-dependent and hidden from the user of the X.500 directory.

Domain Name System (DNS)

Names

A DNS name is an untyped name. A name in DNS consists of an ordered set of strings separated by periods. The strings are ordered left to right, from most specific to most general. The most general strings (top-level Internet domains) are controlled by ANSI and must be one of the following.

Figure 8. Top-level Internet Domains

Domain	Meaning
COM	Commercial organizations
EDU	Educational institutions
GOV	Government institutions
MIL	Military Groups
NET	Major network support centers
ORG	Organizations other than those above
INT	International organizations
Country code	Each country (geographic scheme)

Any suffix to a string is separated by a period. The suffix represents a domain for the string. For example, in the name

```
austin.ibm.com
```

the name component *austin* has the suffix *ibm.com*, so *austin* is in domain *ibm.com*. Further, *ibm* represents a second-level domain, while *com* represents a first-level domain.

In the Directory resource manager federated namespace, DNS names can be expressed as LDAP distinguished names (DN). The DN is part of the LDAP URL. An example of this is

DNS style: development.endicott.ibm.com

expressed as

LDAP DN: `DC=development,DC=endicott,DC=ibm,DC=com,O=Internet`

LDAP URL: `ldap://ibm.com/DC=development,DC=endicott,DC=ibm,DC=com,O=Internet`

Note: DNS style names that are expressed as distinguished names are subject to change per the IETF. The current rule is that each component of a DNS name becomes an attribute of type DC (domain component) and `O=Internet` is appended.

Namespace

The DNS namespace is hierarchical. Each second-level domain is located below a first-level domain. There can be any number of levels of domains. Each name represents an entry in DNS. Each entry can contain a small number of attributes. The syntax of these attributes is well defined. A subset of the namespace is shown in Figure 9 on page 23.

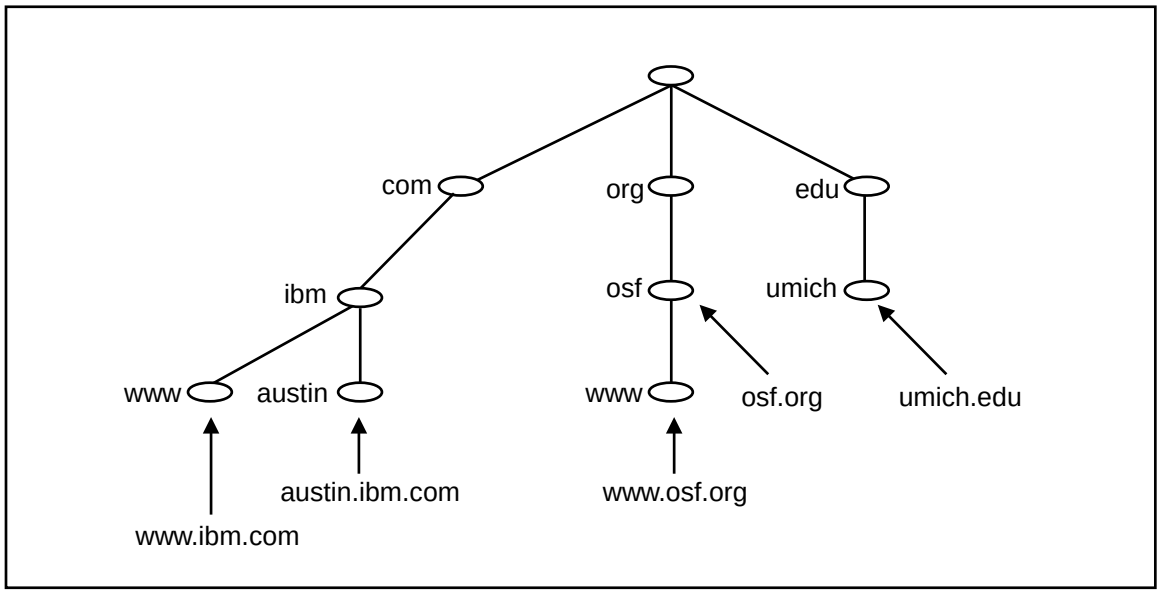


Figure 9. DNS Namespace

Structure

The general structure of a DNS Directory Service is shown in Figure 10.

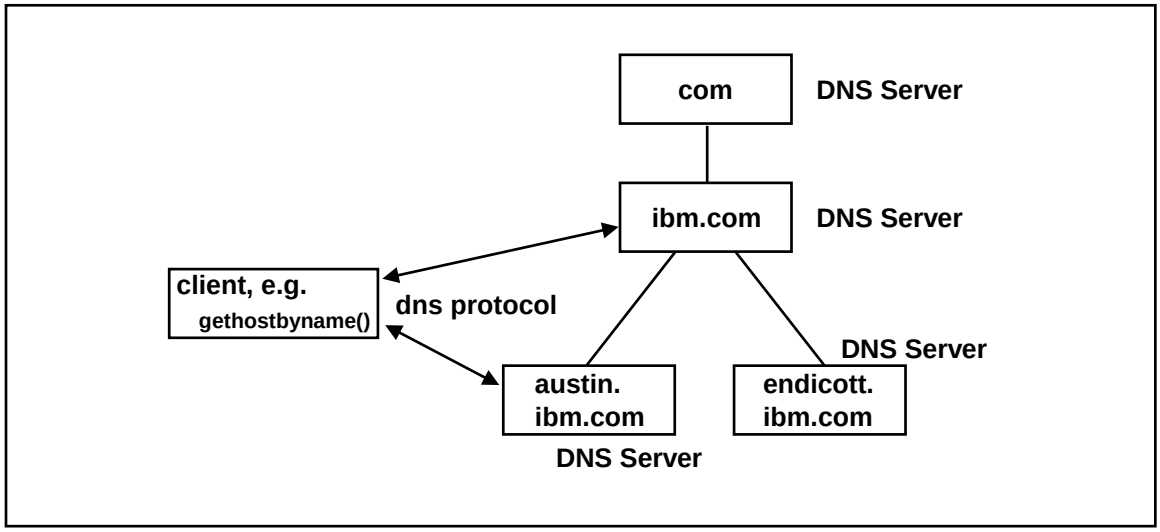


Figure 10. General Structure of a DNS Directory Service

Client: Access to information in DNS is defined by a socket-level protocol used to communicate with a DNS server. Typically, C language calls are provided by the TCP/IP implementation. These calls handle management of the socket interface. An example C language call that accesses DNS is *gethostbyname()*, which, given a DNS name, returns the IP address of the host represented by the name. Other interfaces can access different information stored by the DNS server.

Server: The directory server in the Domain Name System is the DNS Server. The DNS servers accept requests from clients (that use the library calls for DNS) and other DNS servers. Communication is via the DNS protocol. Each DNS server manages a partition of the DNS namespace.

Partitioning: DNS servers can be set up such that the namespace is partitioned across multiple DNS servers. During a name resolution, a server determines the next DNS server to contact if it does not contain the information for the entry. This is done based on the name of the entry requested and the domain name that the DNS server manages.

Replication: While multiple DNS servers can serve the same entries in the DNS namespace, no replication or consistency is maintained across servers. Servers do employ caching to allow for performance improvements on DNS queries. A client can choose to use the cached copy or request the actual entry.

Name Resolution: Two basic methods are used for name resolution in DNS: chaining and referral. With chaining, the DNS server acts on behalf of the client. If the name requested is not contained in the domain the DNS server manages, the server's domain server (one level higher in the name hierarchy) is contacted. If the name requested is contained in the domain but not in the DNS server, a subdomain server is contacted. With referral, the server contacted returns the address of the next DNS server to contact if it does not contain the information requested. The client must then contact the "referred to" DNS server.

Information Model: The information model for DNS directory information is small and fixed. The attributes supported in DNS entries are shown in Figure 11.

Figure 11. DNS Resource Record Types

Type	Meaning	Contents
A	Host address	32-bit IP address
CNAME	Canonical name	Canonical domain name for an alias
HINFO	CPU and OS	Name of CPU and operating system
MINFO	Mailbox information	Information about a mailbox or mail list
MX	Mail exchanger	16-bit reference and name of host that acts as mail exchanger for the domain
NS	Name server	Name of authoritative server for domain
PTR	Pointer	Domain name (like a symbolic link)
SOA	Start of authority	Multiple fields that specify which parts of the naming hierarchy a server implements
TXT	Arbitrary text	Uninterpreted string of ASCII text

The fixed nature of the information model for DNS makes it hard to use as a generalized directory service.

Schema: Each entry in the DNS namespace can have attributes taken from the set of attributes described in the information model. These are not required but are the only ones available.

Directory Information Tree and Directory Information Base: The method of storage for the DIT of an DNS directory is implementation-dependent and hidden from the user of the DNS directory.

Distributed Computing Environment (DCE)

Names

DCE defines domains called cells that must be cataloged in the global namespace (X.500 or DNS).

The structure of a DCE name is:

```
/.../<cell_name>/<cell-relative_name>
```

The left-most element of any valid DCE name is the root prefix "/...". This is the global root. It signifies that the immediately following elements form the name of a global namespace entry. Usually, the entry's contents consist of binding information for a cell, and the name of the global entry is the name of the cell.

Following the root prefix, the DCE name specifies the cell name. The cell name will be a typed name if X.500 is used and untyped if DNS is used. Example names in DCE are:

```
/.../c=US/o=IBM/ou=Marketing/ou=cell1/subsys/dce

/.../cell1.marketing.ibm.com/subsys/dce
```

After the cell name, the next component of a name is the cell-relative name; it can contain namespace junctions. Names that span naming systems are connected together via junctions. A junction is an entry in the namespace (a leaf entry in the superior naming system) that contains information on how to contact the next naming system for name resolution. There is no way by looking at the name to know which name component is a junction. Based on Figure 12 on page 26, examples of junctioned names in DCE are:

- Registry, located at `/.../ibm.com/sec/principal/ziggy`
- Distributed File System, located at `/.../ibm.com/fs/usr/ziggy`
- Host Data, located at `/.../ibm.com/hosts/<hostname>/config/srvrconf/video_clip`

In the Open Blueprint Directory resource manager federated namespace, DCE names can be expressed as LDAP distinguished names (DN). The DN is part of the LDAP URL. An example of this is:

DCE style `/.../c=US/o=IBM/ou=Marketing/ou=cell1/subsys/dce`

expressed as

LDAP DN `container=dce,container=subsys,ou=cell1,ou=Marketing,o=IBM,c=US`

LDAP URL `ldap://ibm.com/container=dce,container=subsys,ou=cell1,ou=Marketing,o=IBM,=US`

Note: DCE style names expressed as distinguished names are subject to change given pending IETF discussions regarding typeless names in the LDAP.

Namespace

Figure 12 on page 26 shows a DCE namespace layout. /... is the root of the global namespace. ibm.com is a cell named in the DNS namespace. The cell name could alternatively be of the form `C=US/O=IBM` where the cell named is in the X.500 namespace. Each cell has the basic layout shown in Figure 12 on page 26. Each of the entries below the cell name are in the cell namespace; some are junctions to other namespaces.

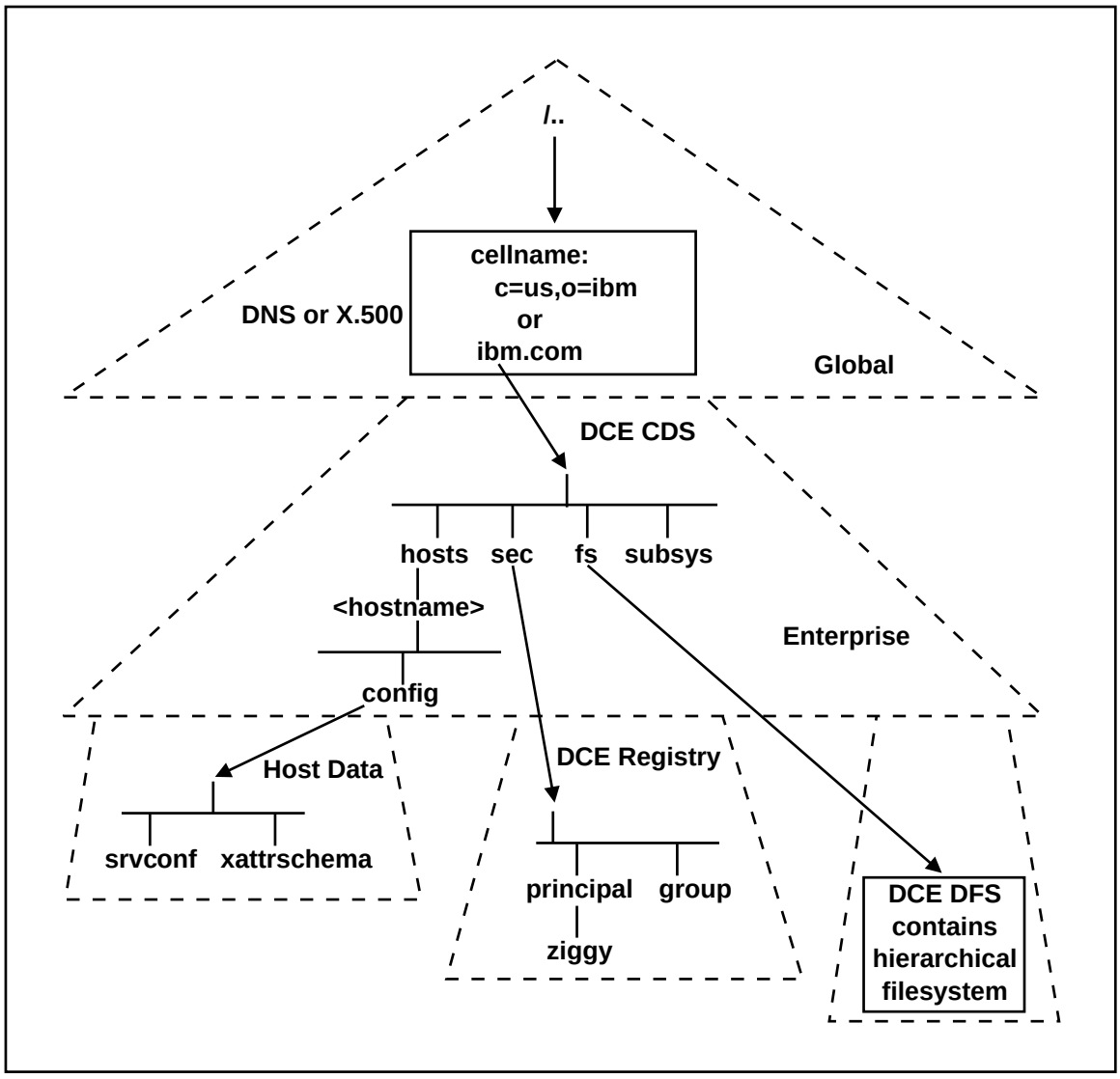


Figure 12. DCE Namespace

The cell also supports the following namespaces, each of which is a junction in the DCE Cell Directory Service (CDS) namespace:

- `hosts/<hostname>/config`

  The config namespace contains information pertinent to that <hostname> being part of the cell such as servers configured for that system, servers currently executing on that system, and schema definitions.

- `sec`

  This is a junction to the security namespace that contains information such as the definitions of users, groups, and security policies for the cell.

- `fs`

  This is a junction to the distributed file system namespace.

Structure

The general structure of the DCE Directory Service is shown in Figure 13 on page 27.

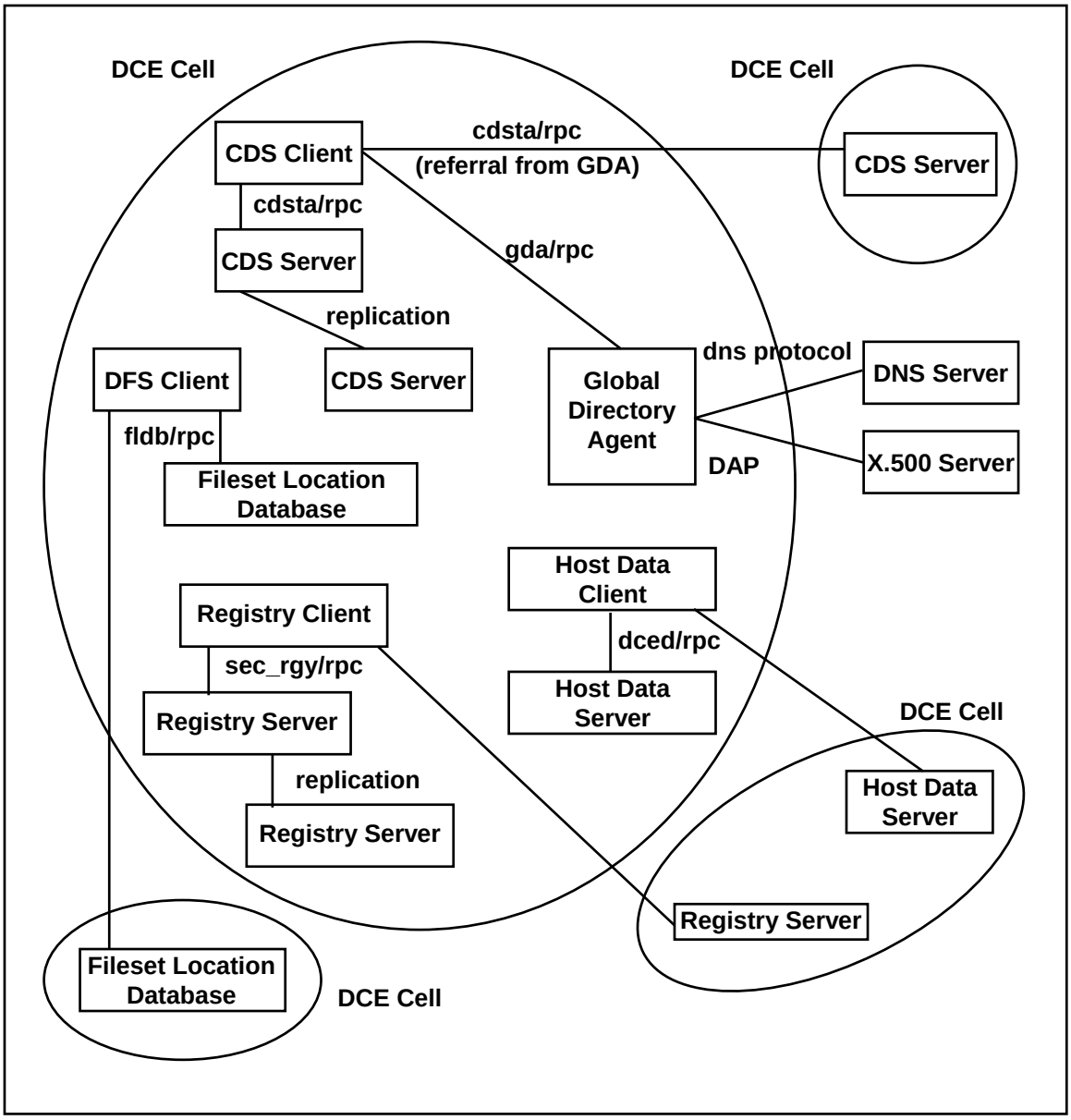


Figure 13. General Structure of the DCE Directory Service

Client: Access to information stored in a DCE directory service is accessible through a DCE directory's client. There are clients for the Cell Directory Service (CDS), the Registry, the Host Data database, and the Fileset Location Database (FLDB). These clients communicate with one or more corresponding DCE directory servers to locate and extract the information from the directory information database (DIB). Each client communicates with its corresponding server through its own protocol. These protocols are transmitted through DCE RPC.

The client function implements the following programming interfaces to access information stored in the DCE directory services:

- X/Open Directory Services/ X/Open Object Management (XDS/XOM)

  This is a procedural interface defined by X/Open that includes operations for name look-up/resolution, bind, unbind, link, attribute operations, and search. It is used to access information in the CDS and X.500.

- Name Service Independent API (NSI)

  This is a procedural interface defined in the X/Open Remote Procedure Call specification that provides binding capabilities in conjunction with a search capability using RPC group and RPC profile entries defined in the directory service. This interface allows existing RPC-based applications to be supported in the Open Blueprint. This interface is also used by the Open Blueprint Distribution Services. This interface is used to access information in the CDS.

- X/Open DCE Security Services sec_rgy*()

  This interface is used to access information in the Registry. This is a procedural interface defined in the X/Open DCE Security Services specification which provides access to information in the Registry, such as information on DCE principal ziggy.

- DCE Host Data

  This is a procedural interface defined in the Open Software Foundation (OSF) DCE specification that provides access to information in the Host Data namespace, such as which servers are configured and running on what hosts. This interface also allows a schema to be defined for this information.

- DCE DFS

  DCE does not define a programming interface to the file system. Instead, access is via the typical file system interfaces and commands such as fopen() and ls.

Server: Each DCE Server (CDS, Registry, Host Data, FLDB) runs on a node containing a database of directory information. It responds to queries from clients by accessing its databases. Communication is via remote procedure calls (RPCs).

Partitioning: To allow for concurrent access, load balancing, and availability, each DCE server stores a partition of the DCE namespace. Each server manages a partition of the DCE namespace. These partitions represent the master copy (read/write) in the DCE namespace. Sub-trees can be defined under sub-trees and, as a consequence, form a hierarchy. The CDS, Host Data, and FLDB partitions can be distributed across one or more servers. The Registry cannot itself be partitioned, and therefore, is contained on a single server.

Replication: Replication is used to maintain more than one copy of information in the DIB. This is usually done for availability purposes. A DCE directory server can replicate information it stores to other DCE servers of the same type. Each DCE server has its own protocol for replicating information. The level of granularity for replication is an entry for CDS, a file set for FLDB, and the entire Registry; no replication of the Host Data is possible. A master/copy approach is used in which the information for an entry stored at one server is deemed the master version of the information. If an update is made to the information, all other copies of the information are modified, possibly, at a later time. Updates to copies of information can be immediate or periodic.

Caching is also a form of replication. The DCE clients can cache information so that information will be available on the local node rather than having to repeatedly query directory servers.

Name Resolution: The referral method is used for name resolution in the CDS. Referral is used with names that contain junctions and for intercell access.

Resolution of junctions: If the CDS server that a CDS client contacted cannot resolve the name, the unresolved portion of the name is returned to the client so the client can read the junction (or referral information) contained in the namespace entry where resolution stopped to determine another server (in the junctioned namespace) that can be contacted to further resolve the name.

Intercell: Access from one cell to resources in another cell is called intercell communication. The Global Directory Agent (GDA) helps establish communication (transparently) between cells. It takes a name that

cannot be found in the local cell and finds the foreign cell in which the name resides, using X.500 or DNS, depending on where the foreign cell is registered. When a CDS server cannot resolve the name because the cell name addressed is outside that CDS server's cell, the CDS server refers the CDS client to the GDA for cell name resolution. After the foreign cell name is resolved, the GDA sends a referral to the CDS client to finish resolution via the CDS server in the foreign cell.

Information Model: The information model for DCE directory information is modifiable and extendable. An information model exists for each DCE directory; these base models are required. Information is stored in directory entries. Attributes are type=value pairs associated with a directory entry. CDS attribute types are defined by an object identifier (OID); the value has a defined syntax. Registry, Host Data, and FLDB attribute types are defined by universal unique identifiers (UUIDs). Attributes can be single-valued or multi-valued. An entry can contain multiple attributes. Entries are organized hierarchically.

Schema: Each DCE directory has a defined schema that is DCE-specific; however, the schemas are extendable to allow for non-DCE objects and attributes. The DCE directories do not enforce a schema (checking for mandatory/optional attributes and where a type of entry can be placed in the hierarchy) when entries and attributes are created.

Directory Information Tree and Directory Information Base: The method of storage for the DIT of a DCE directory is implementation-dependent and hidden from the users of the DCE directories.

Lotus Notes Name and Address Book

The Lotus Notes Name and Address Book is really a Lotus Notes database. As a Lotus Notes database, the Name and Address Book contains Lotus Notes documents. Each document contains a set of name-value pairs. Unlike other directory services, the set of formats for the attributes of a document is a much richer set. For example, a "rich text format" attribute-type exists that allows images and formatted text information to be stored in a name-value pair.

Names

Names in the Name and Address Book are values of the name attribute in each Name and Address Book document. The format of this name approximates an X.500-like format, though a number of formats are supported.

A Notes name is (from the user's perspective) typeless - only the value is generally used in a name, for example, Joe Q. Public. However, a type is associated with the name, such as cn for common name. So the usable shortened form of a name might be:

```
Joe Q. Public/Austin/IBM/US
```

and the typed form of the name would be:

```
cn=Joe Q. Public/ou=Austin/o=IBM/c=US
```

The order of name components in a Notes name is reversed from the X.500 style names. In Notes, the most specific name component appears first, followed by the less specific name component moving up the DIT hierarchy.

In the Open Blueprint Directory Service federated namespace, Name and Address Book names can be expressed as LDAP distinguished names (DN). The DN is part of the LDAP URL. This mapping is most easily made from the typed form of the name. An example of this is:

Address Book style: `Joe Q. Public/ou=Austin/o=IBM/c=US`

expressed as

LDAP DN: `cn=Joe Q. Public,ou=Austin,o=IBM,c=US`

LDAP URL: `ldap://ibm.com/cn=Joe Q. Public,ou=Austin,o=IBM,c=US`

Namespace

The Notes Name and Address Book namespace appears hierarchical (see “Information Model” on page 31). Each entry contains attributes. Attributes are typed and multiple values can be stored for a given attribute. There is no ordering to the multiple values. Each attribute type conforms to an attribute syntax. The namespace consists of all values possible for the name attributes in the Name and Address Book document.

Structure

The Notes general structure with Name and Address Books is shown in Figure 14 on page 31.

Client: Information stored in a Notes Name and Address Book is accessible through a Notes client. The Notes client communicates with one or more Notes Servers to locate and extract the information from the directory information database (DIB) in the Name and Address Book. Notes clients communicate with Notes servers with a Notes-defined protocol.

A Notes client has a number of different interfaces including graphical-user-interface-based programs and C language callable programming interfaces. The commonly used programming interface is Lotus Script. This interface provides methods for accessing and modifying information stored in Notes databases. Java is also supported.

A Personal Address Book is stored at the client as a Notes database.

Server: The Notes server accept requests from Notes clients and other Notes servers. Notes servers communicate with each other and with Notes clients using a Notes-specific protocol.

Partitioning: Each Notes server in a domain stores a complete replica of that domain's Name and Address Book.

Replication: Replication is used to maintain more than one copy of information in the Notes Name and Address Book. This is usually done for availability. A Notes server can replicate information it stores to other Notes Servers. Notes servers replicate information using a Notes-specific protocol. All copies of Notes Name and Address Books are read/write. An update can be made to any copy; all other copies of the information are modified at a later time. Updates to copies of information can be immediate or periodic.

Name Resolution: Two basic methods are used for name resolution in Notes, referral and chaining. With referral, if the Notes client cannot resolve the name first in the user's Personal Name and Address Book, the Notes client refers the request to the Notes server for resolution. Notes Name and Address Books can be chained so that if a request cannot be satisfied by the current Notes server, that server can contact another Notes server to check the Address Book that it manages for resolving the request, returning a result to the Notes client only after the chained request is complete.

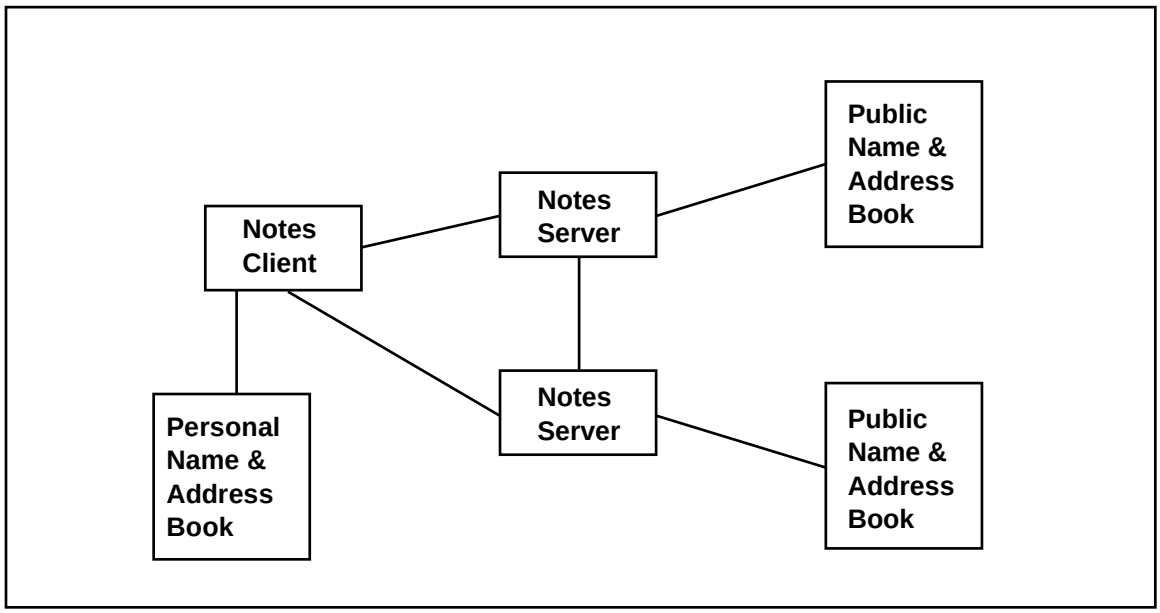


Figure 14. General Structure of Notes with Name and Address Books

Information Model: The information model for the Notes Name and Address Book information is modifiable and extendable. Information is stored in an address book into entries that contain attributes. Attributes are (possibly multi-valued) name-value pairs, and the value has a defined syntax. An entry can contain multiple attributes. The organization is flat although a name may appear to be hierarchical. For example, the name

```
cn=Joe Q. Public/o=ibm/c=us
```

appears to be a hierarchical name, but to Notes the entry (name-value pair) is

```
name ="cn=Joe Q. Public/o=ibm/c=us"
```

Schema: A Notes schema defines rules for names and what attributes an entry can contain. A Notes schema is required, and it can be extended. However, the attributes associated with a user name are fixed and cannot be modified. The schema is enforced. The schema is defined by the set of database forms stored in the Name and Address Book.

Directory Information Tree and Directory Information Base: The method of storage for the DIT of a Lotus Notes directory is a Notes database. Its design is viewable and modifiable by Notes users.

LDAP

Names

The LDAP URL uses the scheme *ldap* and has the form:

```
ldap://<host>:<port>/<path>
```

where <path> has the form: `<dn>[?<attributes>[?<scope>?<filter>]]`

The <dn> is an LDAP distinguished name using the string representation. The <attributes> indicates which attributes should be returned from the entry or entries. If omitted, all attributes for entry or entries are returned. The <scope> specifies the scope of the search to perform. Scopes can be the current entry, 1-level (current entry's children), or whole subtree. The <filter> specifies the search filter to apply to entries within the specified scope during the search.

The URL allows Internet clients such as Web browsers to have direct access to the LDAP protocol. Examples of LDAP URLs are:

- `ldap://austin.ibm.com/o=ibm,c=US`

  This LDAP URL corresponds to a base object search of the `o=ibm, c= US` entry using a filter of (`objectclass=*`), requesting all attributes.

- `ldap://austin.ibm.com/o=ibm,c=US?postalAddress`

  This LDAP URL refers to only the postalAddress attribute of the IBM entry

- `ldap:///o=ibm,c=US??sub?(cn=Joe Q. Public)`

  This LDAP URL refers to the set of entries found by querying any X.500-capable LDAP server and doing a subtree search of the IBM subtree for any entry with a common name of `Joe Q. Public`, retrieving all attributes.

The string representation for names has the form <type=value> for single valued RDNs. The plus sign (+) is used to form multi-valued RDNs, <type=value> + <type=value>. RDNs are separated by commas. Examples in string form are:

- `cn=Joe Q. Public,o=ibm,c=US`
- `ou=Austin + cn=Joe Q. Public,o=ibm,c=US`

The order of RDNs in an LDAP name is reversed from the X.500. In LDAP, the most specific RDN appears first followed by the less specific RDNs moving up the DIT hierarchy.

This string representation is used for the distinguished name (dn) portion of the URL and for names when using the LDAP programming interface.

Namespace

The namespace is hierarchical. Each entry contains attributes. Attributes are typed, and multiple values can be stored for an attribute. There is no ordering to the multiple values. Each attribute type conforms to an attribute syntax. Examples of syntaxes include PrintableString and Integer. ASN.1, a standard type declarative language, is used to describe attributes stored in the namespace. Aliases are supported as in X.500.

The LDAP namespace is a logical loosely-federated namespace (see Figure 5 on page 17). Each entry in each directory service namespace can be expressed using the LDAP string format. Hence, the federation is presented as a uniform namespace with entries expressed as LDAP distinguished names. In the Open Blueprint Directory resource manager entries are in existing namespaces, for example CDS, so the entry is also known by that namespace's name syntax.

Structure

The structure of the LDAP Directory Service is shown in Figure 15 on page 33. An LDAP server is a server that accepts the LDAP protocol. The LDAP server supports one or more directory services, for example, CDS. Clients continue to communicate with their respective servers, for example, CDS clients with CDS servers and LDAP clients with LDAP servers. Replication between like directory servers continues to work.

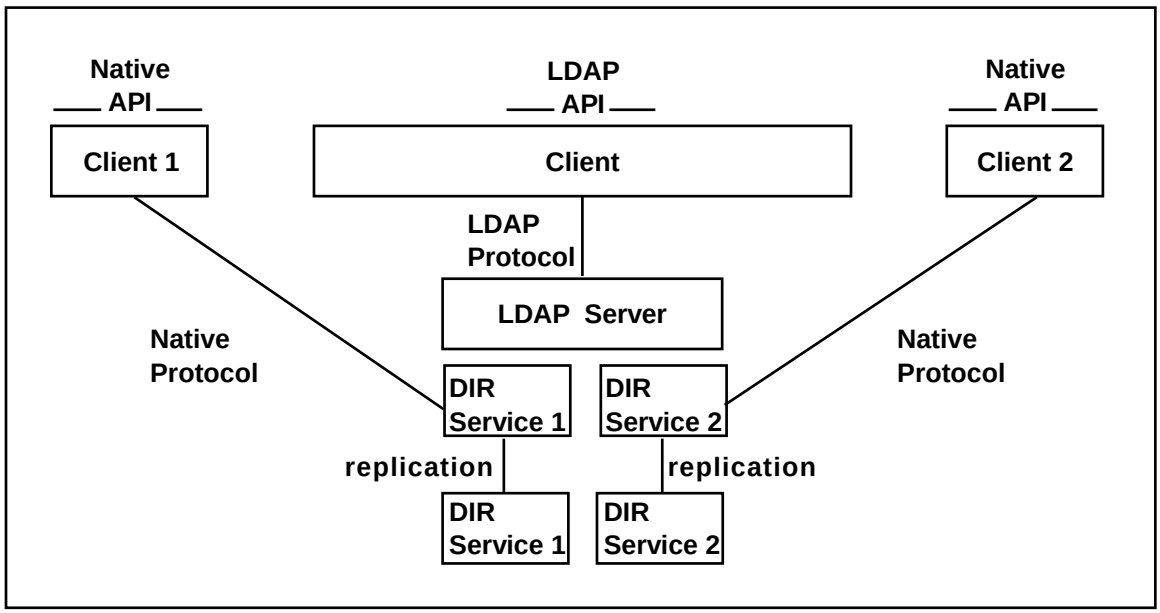


Figure 15. General Structure of the LDAP Directory Service

Client: Access to information stored in the LDAP directory service is accessible through an LDAP client and the directory service's native client. The LDAP client communicates with one or more directory servers that accept the LDAP protocol to locate and extract the information from the directory information database (DIB). The client can present the LDAP interface in a number of ways. Examples include the LDAP URL, the LDAP C programming interface, Java class libraries, or as an application such as a graphical user interface.

Server: An LDAP server is any directory server that accepts the LDAP protocol. The LDAP server in Figure 15 supports multiple back-ends.

Note: The LDAP server is based on the University of Michigan reference implementation of Stand-Alone LDAP Daemon (slapd) that supports multiple backends.

Partitioning: The LDAP namespace can be thought of as partitioned. However, partitions are stored in different directory services. Partitioning is based on whole subtrees. Subtree partitions can appear below other subtree partitions. The DCE Cell Directory namespace and the DCE Registry namespace are examples of partitions within the LDAP namespace. DCE Registry is also an example of a partition that appears below another partition, that is, the DCE Cell Directory namespace.

Replication: Replication between like directory services is provided by the existing directory servers.

Name Resolution: Two basic methods are used for name resolution in LDAP: referral and chaining. With referral, if the LDAP server that an LDAP client contacts cannot resolve the name, the unresolved portion of the name is returned to the LDAP client along with a URL of another LDAP server that can be

contacted to further resolve the name. With chaining, the LDAP server can contact another LDAP server on the LDAP client's (and thus the user's) behalf, returning a result to the LDAP client only after the chained request is complete.

Information Model: The information model is a subset of the X.500 information model, but is extendable and modifiable. Complex attribute types supported in the X.500 are not supported in the LDAP. As in the X.500 directory service, information is stored into entries that contain attributes. Attributes are (possibly multi-valued) `<type=value>` pairs in which the type is defined by an object identifier (OID); value has a defined syntax. An entry can contain multiple attributes. Entries are organized hierarchically by their distinguished name. Entries whose distinguished name contains the distinguished name of another entry as a suffix are considered to reside under the latter entry in the hierarchy.

Schema: The Internet Draft, *Lightweight Directory Access Protocol: Standard and Pilot Attribute Definitions*, defines a set of standard attributes that all LDAP servers must recognize. The set of attribute types supported is extendable. A schema can be created with these attribute types and optionally enforced. In the Open Blueprint Directory resource manager, the LDAP schema is mapped to the schema of the existing directory service, where applicable. Schema enforcement is subject to that provided by the underlying directory service.

Directory Information Tree and Directory Information Base: An LDAP server makes use of the DIBs that the underlying directory services provide. Thus, the DIB can have many different forms in an LDAP server.

LDAP and Meta-Directory Usage Example

Figure 16 on page 35 shows an example of how a federated namespace might look in a customer environment.

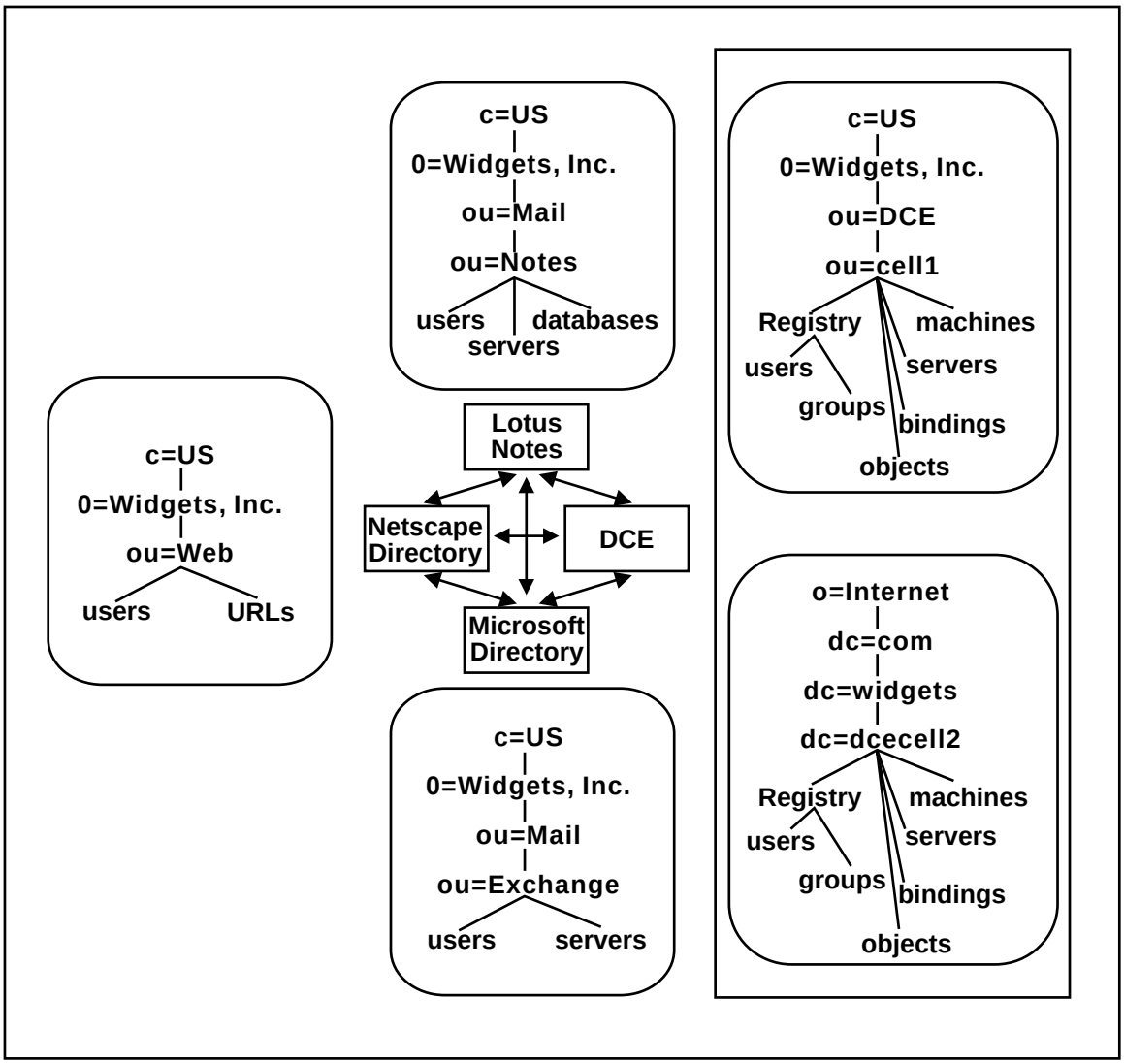


Figure 16. Example of the Open Blueprint Directory Namespace for Company Widgets, Inc.

Some of the benefits of the Open Blueprint Directory resource manager federation of namespaces using the LDAP protocol are shown through the examples in Figure 16 and Figure 17 on page 36. Consider a mythical company called Widgets, Inc. This company is of moderate size and is at the cutting edge of computing technology. Across the corporation, many departments have enabled themselves for maximum productivity within the group. Unfortunately, at the corporate level, much time is lost communicating information between groups. As it happened, each department chose the enabling technology that best fit its needs.

Some departments chose intranet browser technology, making use of a Netscape Directory Service to store information about its users as well as information about interesting URLs.

Other departments have found that the Microsoft Exchange mail product works well for their communication needs and have invested in setting up this environment, defining users, mail addresses, and POP servers.

Still other departments have a strong need for collaborating on documents and have chosen to use Lotus Notes. Because of Notes' mail capabilities, these departments also use Lotus Notes for e-mail.

Some of the development departments have found the DCE distributed file service (DFS) to be a superior way of file sharing between development workstations. In addition, the flexibility and extensibility of the DCE environment with security features built into its communication model allow these departments to quickly create secure applications for their own use.

The result is a set of namespaces that each hold critical information for the respective departments that use them but are inaccessible by other departments across the corporation. The Open Blueprint Directory resource manager namespace is the federation of these separate namespaces. By federating them together, information in all of these directory services can be accessed and used. The Open Blueprint Directory resource manager brings the separate directory services together under a common naming model. Here are some examples:

Figure 17. Examples of Names in the Widget, Inc. Federal Namespace

Specific Namespace Name	**Directory Federated**	**Namespace (LDAP) Name**
cn=Joe Q. Public, ou=Web, o="Widgets, Inc.", c=US	Netscape	cn=Joe Q. Public, ou=Web, o="Widgets, Inc.", c=US
cn=Netscape, ou=Home Pages, ou=Web, o="Widgets, Inc.", c=US	Netscape	cn=Netscape, ou=Home Pages, ou=Web, o="Widgets, Inc.", c=US
Jane P. Public	Microsoft	cn=Jane P. Public, ou=Exchange, ou=Mail, o="Widgets, Inc.", c=US
POP Server 1	Microsoft	mailserver=POP Server1, ou=Exchange, ou=Mail, o="Widgets, Inc.", c=US
POP Server 2	Microsoft	mailserver=POP Server2, ou=Exchange, ou=Mail, o="Widgets, Inc.", c=US
Jerry M. Public	Lotus	cn=Jerry M. Public, ou=Notes, ou=Mail, o="Widgets, Inc.", c=US
AUSNOTES (a Notes Server)	Lotus	mailserver=AUSNOTES, ou=Notes, ou=Mail, o="Widgets, Inc.", c=US
Requirements (a Notes database)	Lotus	database=Requirements, ou=Notes, ou=Mail, o="Widgets, Inc.", c=US
/../c=US/o=Widgets, Inc./ou=DCE/ou=cell1/subsys/dce	DCE	container=dce, container=subsys, ou=cell1, ou=DCE, o="Widgets, Inc.", c=US See note #1 below.
/.../dcecell2.widgets.com/sec	DCE	object=sec, dc=dcecell2, dc=widgets, dc=com, o=Internet See note #2 below.
/.../dcecell2.widgets.com/joepublic	DCE	cn=joepublic, object=sec, dc=dcecell2, dc=widgets, dc=com, o=Internet See note #2 below.

Notes:

1. The attribute *container* is arbitrarily defined to aid in expressing untyped names in the LDAP DN syntax. It describes the name component type in DCE terms and is subject to change pending resolution of expressing untyped name in LDAP DN syntax at IETF.
2. The attribute *object* is arbitrarily defined to aid in expressing untyped names in the LDAP DN syntax. It describes the name component type in DCE terms and is subject to change pending resolution of expressing untyped names in LDAP DN syntax at IETF. For DNS style names that are to be expressed as distinguished names, the current rule is that each component of a DNS name becomes an attribute of `type DC` (domain component) and `o=Internet` is appended. This rule is also subject to change by the IETF.

As the table points out, names in the Open Blueprint Directory resource manager federated namespace correspond to names in the separate namespaces. The important point is that all information can be accessed through the federated namespace.

Figure 18 on page 38 shows the incorporation of a meta-directory for Widgets, Inc. This meta-directory has entries that exist inside the Open Blueprint Directory resource manager namespace. The value of the meta-directory is that it can be used to collect attributes about a common object (person, place, thing). For example, user Joe Q. Public is defined in both the Netscape (as `cn=Joe Q. Public`) and DCE (as `joepublic`) namespaces. The meta-directory entry for `cn=Joe Q. Public, ou=Admin, o="Widgets, Inc.", c=US` would contain all the attributes of the union of these two directory services.

The meta-directory provides synchronization services for like attributes about a common object that exists in one or more directory services.

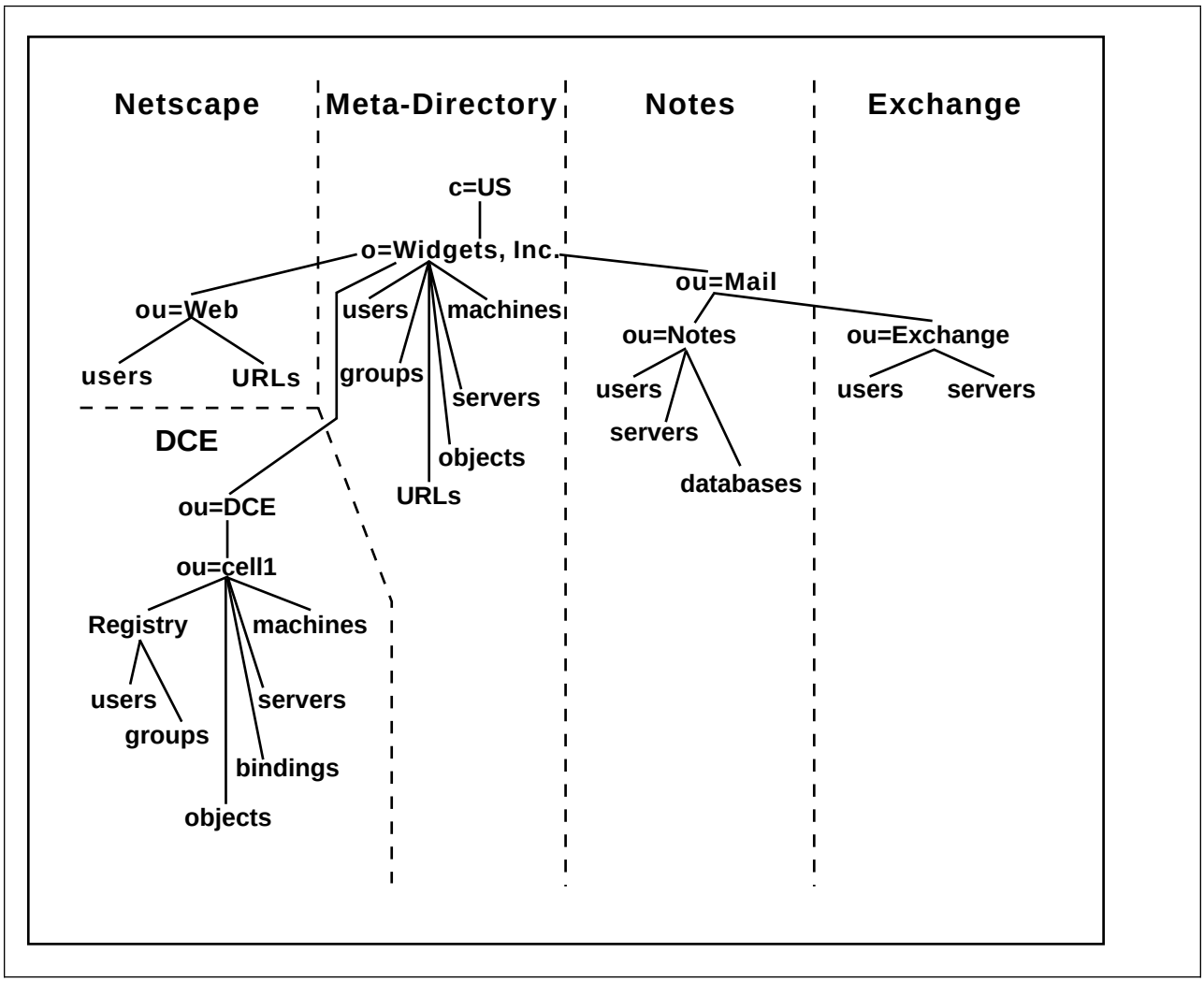


Figure 18. Meta-Directory Applied to Widgets, Inc.

As the example shows, the Open Blueprint Directory resource manager allows information that is scattered across the Widgets, Inc. corporation to be accessed across the corporation. The Open Blueprint Directory resource manager places this data in the federated namespace, positioning it for use from anywhere. The federation of namespaces ties together the enterprise and Internet to allow consistent naming to be used to locate information. This is the goal of the Open Blueprint Directory resource manager.

Relationship to Other Open Blueprint Resource Managers

The Open Blueprint Directory resource manager makes use of several Open Blueprint resource managers. Time services are relied on to maintain synchronized clocks in the network for use in the sequencing of updates to the namespace and for use in replication. Security services are used for authentication to a directory server, access control on directory information, and data integrity (secure communication, confidentiality, packet header, digital signature). RPC is specifically used by the DCE Cell Directory Services and DCE Registry. Open Blueprint System Management is used to manage information stored in the directories and meta-directory.

Appendix A. Bibliography

OSF DCE Application Development Reference, Version 1

OSF DCE Application Development Guide, Version 1.1

OSF DCE Administration Guide, Version 1.1

OSF DCE Command Reference, Version 1.1

Introduction to OSF DCE, Version 1.1

X/Open, *DCE: Directory Services*, Document Number C312, ISBN 1-85912-8-4

X/Open, *DCE: Remote Procedure Call*, Document Number C309, ISBN 1-859-041-5

X/Open, *API to Directory Services (XDS)*, Document Number C317, ISBN 85912-007-5

X/Open, *OSI-Abstract-Data Manipulation (XOM)*, Document Number C315, ISBN 1-85912-008-3

X/Open, *Federated Naming: The XFN Specification*, Document Number P403, ISBN 1-85912-045-8

X.500 family of specifications, 1993 version

Understanding X.500 The Directory, David Chadwick, ISBN 0-412-43020-7

Internet RFC 1034: *"Domain Names - Concepts and Facilities"*

Internet RFC 1035: *"Domain Names - Implementation and Specification"*

Internet RFC 1274: *"The Cosine and Internet X.500 Schema"*

Internet RFC 1738: *"Uniform Resource Locators (URL)"*

Internet RFC 1777: *"Lightweight Directory Access Protocol"*

Internet RFC 1778: *"The String Representation of Standard Attribute Syntaxes"*

Internet RFC 1779: *"A String Representation of Distinguished Names"*

Internet RFC 1798: *"Connection-less Lightweight X.500 Directory Access Protocol"*

Internet RFC 1823: *"The LDAP Application Program Interface"*

Internet RFC 1959: *"An LDAP URL Format"*

Internet RFC 1960: *"A String Representation of LDAP Search Filters"*

Internet Drafts:

- *"Lightweight Directory Access Protocol: Standard and Pilot Attribute Definitions"*, to obsolete RFC 1778
- *"Lightweight Directory Access Protocol (v3)"*, to obsolete RFC 1777, RFC 1798
- *"The String Representation of Standard Attribute Syntaxes"*

- *"A String Representation of LDAP Search Filters"* , to obsolete RFC 1960
- *"The LDAP Application Program Interface"*, to obsolete RFC 1823
- *"Dynamic Updates in the Domain Name System (DNS UPDATE)"*
- *"Deferred Dynamic Updates in the Domain Name System (DNS DEFUPD)"*
- *"Lightweight Directory Access Protocol (v3): UTF-8 "String Representation of Distinguished Names"*
- *"An Approach for Using Domains in LDAP Distinguished Names"*
- *"Lightweight Directory Access Protocol: Extensions for Dynamic Directory Services"*

OMG Name Service Specification, OMG TC Document 93.5.2

Appendix B. Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's product, program, or service may be used. Subject to IBM's valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
USA

Trademarks

The following terms are trademarks the IBM Corporation in the United States or other countries or both:

IBM
IBMLink
MVS
Open Blueprint
RACF

The following terms are trademarks of other companies:

Banyan	Banyan Systems, Incorporated
cc:Mail	cc:Mail, Incorporated
DCE	The Open Software Foundation
Java	Sun Microsystems, Incorporated
Lotus	Lotus Development Corporation
Lotus Notes	Lotus Development Corporation
Microsoft	Microsoft Corporation
Netscape (logo)	Netscape Communications Corporation
Novell	Novell, Incorporated
OSF	Open Software Foundation, Incorporated
SunSoft	Sun Microsystems, Incorporated
X/Open	X/Open Company Limited

Appendix C. Communicating Your Comments to IBM

If you especially like or dislike anything about this paper, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply. Feel free to comment on specific error or omissions, accuracy, organization, subject matter, or completeness of this paper.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

- If you prefer to send comments by FAX, use this number:

  United States and Canada: 1-800-227-5088.

- If you prefer to send comments electronically, use one of these ID's:
  - Internet: **USIB2HPD@VNET.IBM.COM**
  - IBM Mail Exchange: **USIB2HPD at IBMMAIL**
  - IBMLink: **CIBMORCF at RALVM13**

Make sure to include the following in your note:

- Title of this paper
- Page number or topic to which your comment applies

Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

GC23-3915-01