DCE Directory Services

The DCE directory services provide access for applications and users to a federation of naming systems at the global, enterprise, and application levels.

by Michael M. Kong and David Truong

The directory services of the Open Software Foundation's Distributed Computing Environment (DCE) enable distributed applications to use a variety of naming systems, both within and outside a DCE cell. Naming systems make it possible to give an object a textual name that is easier for humans to use—easier to read, pronounce, type, remember, and intuit—than an identifier such as a DCE universal unique identifier (UUID). Information about the locations of objects can be stored in naming systems so that users can access an object by name, with no a priori knowledge of its location.

DCE provides three directory services:

- The Global Directory Service (GDS) is the DCE implementation of the CCITT (International Telegraph and Telephone Consultative Committee) 1988 X.500 international standard. GDS is a distributed, replicated directory service that manages a global namespace for names anywhere in the world.
- The Cell Directory Service (CDS) is a distributed, replicated directory service that manages names within a DCE cell.
- The Global Directory Agent (GDA) is a daemon that uses global name services to help applications access names in remote cells. GDA interacts with either X.500 services such as GDS or Internet Domain Name System (DNS) services such as the Berkeley Internet Name Domain (BIND) name server, named.

Through these services, DCE applications can access several interconnected namespaces, including X.500, DNS, CDS, the DCE security namespace (see article, page 41), and the DCE Distributed File Service (DFS) filespace (see article, page 6).

The DCE Namespace

The DCE namespace is a federation of namespaces at three levels: global namespaces, an enterprise namespace, and application namespaces. A DCE name can span one, two, or all three of these levels. GDS and BIND name servers provide X.500 and DNS global namespaces in which the names of DCE cells are stored. Within each DCE cell, CDS provides an enterprise namespace, and the names of CDS objects in that namespace are relative to that cell. At the application level, DCE subsystems including DCE security and DFS define their own namespaces.

DCE names are hierarchical names consisting of a series of components delimited by the / character. The first component of a DCE name is one of three prefixes denoting the root of a namespace:

- /... is the global root. A name that begins with /... is called a *global name*.
- /.: is the root of the local cell. This prefix is shorthand for /.../<local-cell-name>. Names that begin with /.: are called *local names*.
- /: is the root of the DFS filespace. This prefix, shorthand for /.:/fs, makes DFS filenames easier to use.

Within the local DCE cell, local and global names for an object are equivalent and interchangeable. However, when a user references a local name, the resolution of the name is relative to whatever cell the user is in. Hence, to access an object in a remote cell, a user must refer to the object by its global name.

A DCE cell has a global name that may be either an X.500 name stored in GDS or a DNS name stored in a BIND database. For example, a cell at HP's Cupertino site could have the GDS global name /.../c=us/o=hp/ou=cupertino. In X.500 syntax, the components of a name are separated by the / character, and each component describes an attribute of the object. The component o=hp, for instance, signifies the organization named HP. The DNS global name /.../cssl.cell.ch.hp.com might name a cell in HP's Chelmsford systems software lab (CSSL). As this example shows, a DNS name is a single component of a DCE name but is itself a compound name. DNS names are made up of names in the hierarchical DNS namespace, separated by periods and ordered right-to-left from the DNS root.

Objects in a cell have names that are composed of the cell name, a CDS name, and possibly a name from an application namespace. Some objects, such as RPC servers, are named directly in the CDS namespace; their names consist of a cell name plus a CDS name. Other objects, such as DFS files or DCE security principals, are managed by a service that implements an application namespace. The name for such an object is formed by concatenating a cell name, a CDS name for the root of the application namespace, and an application name relative to that root. For example:

- If the HP Chelmsford systems software lab cell is running an RPC server from the Acme Database Company, that server might be registered under the name /.../cssl.cell.ch.hp.com/ acme/acme_server (see Fig. 1). Within the CSSL cell, /.:/acme/acme_server would be an equivalent name for the server.
- If the HP CSSL cell includes a principal named Nijinsky, that principal would have the global name /.../cssl.cell.ch.hp.com/sec/principal/nijinksy (see Fig. 2) and the local name /.:/sec/principal/nijinksy.
- If the DFS filespace in the HP Cupertino cell contains a file called /users/sergey/mail/igor, that file would have the global

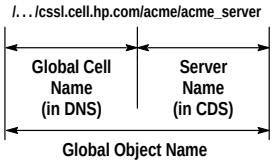

Fig. 1. A DCE global name for a server.

name /... /c=us/o=hp/ou=cupertino/fs/users/sergey/mail/igor (see Fig. 3). Within the Cupertino cell, the names /.:/fs/users/sergey/mail/igor and /:/users/sergey/mail/igor would be equivalent names for the file.

Directory Service Interfaces

DCE offers two sets of programming interfaces to directory services. The RPC Name Service Independent (NSI) API is a generic naming interface provided by DCE RPC. The X/Open® Directory Service (XDS) and X/Open OSI Abstract Data Manipulation (XOM) APIs are interfaces based on CCITT X.500 standards.

Transparently to an application, the DCE directory services interact with each other as necessary to resolve names. Fig. 4 illustrates some of the interrelationships between these services.

When a DCE application passes a name to the RPC NSI API, CDS client software (in the DCE run-time library and in CDS client daemons) uses DCE RPC to contact a CDS server either in the local cell or in a remote cell to look up the name. Names in the local cell are passed directly to a CDS server in the cell. Names in a remote cell are passed to a GDA daemon, which performs a lookup in X.500 or DNS, depending on the syntax of the cell name, to obtain the location and identity of a CDS server in the remote cell. The CDS client software then uses this information to contact the remote CDS server.

When an application passes a name to the XDS/XOM APIs, the XDS code in the DCE run-time library resolves the name according to its syntax. If the name consists purely of components such as c=us and o=hp, the XDS library passes the name to the GDS client code, which contacts the GDS server to look up the name. If any portion of the name is not in GDS syntax, the name is passed to the CDS client code and resolved in the same way as names passed through the RPC NSI API.

GDS Directory Structure

The GDS directory is a collection of information about objects that exist in the world. Information about objects is stored in a database called a *directory information base*. A directory information base contains an entry that completely

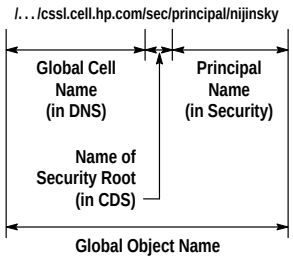

Fig. 2. A DCE global name for a principal.

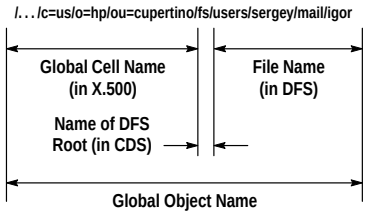

Fig. 3. A DCE global name for a file.

describes each object and may also contain an alias entry that provides an alternative name for an object entry.

An entry in the directory information base consists of a set of attributes, each of which stores information about the object to which the entry refers. An attribute is made up of an attribute type and one or more attribute values. For example, an entry for a person might include attributes whose attribute types are surname, common name, postal address, and telephone number. Attributes that have more than one value are called multivalued attributes. A person with more than one telephone number would have a multivalued telephone number attribute.

Each entry can belong to one or more object classes. An object class of an entry restricts the permitted attributes for that entry. The mandatory and optional attributes of entries in an object class are determined by object class rules, and

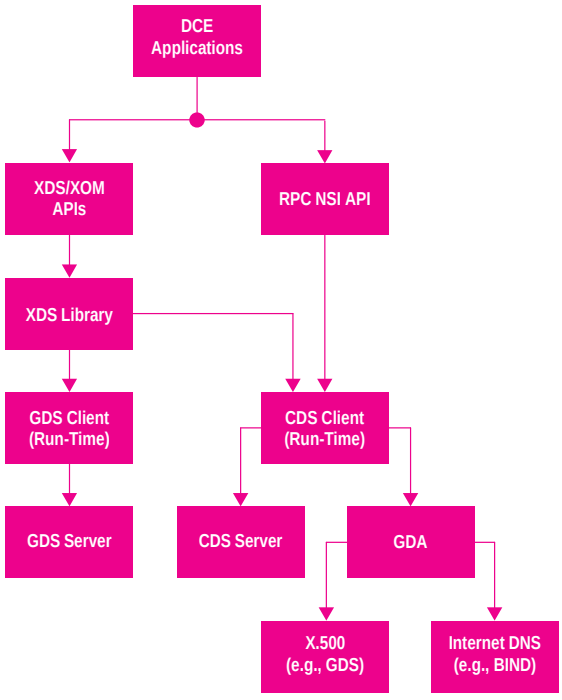

Fig. 4. Interrelationships between DCE directory services. The application program interfaces (APIs) are the DCE remote procedure call name service independent API (RPC NSI API) and the X/Open Directory Service (XDS) and X/Open OSI Abstract Data Manipulation (XOM) APIs. GDS is the DCE Global Directory Service and CDS is the DCE Cell Directory Service. GDA is the DCE Global Directory Agent. X.500 is an international standard implemented by the DCE GDS. DNS is the Internet Domain Name System. The Berkeley Internet Name Domain (BIND) is an implementation of DNS.

these rules are part of a schema. For example, an entry representing an organization must contain an attribute called Organization-Name, which has the name of the organization as its value. An entry can contain optional attributes that describe the organization: the state or locality in which the organization resides, the postal address of the organization, and so on. As a general rule, all entries must contain the Object-Class attribute, which contains the list of object classes to which the entry belongs. If an entry belongs to more than one object class, all object classes must be listed in this attribute.

As discussed above, attribute types and object classes have human-readable names that are meaningful and unique, but they are not used in the protocols; an object identifier is used instead. An object identifier is a hierarchical number assigned by a registration authority. The possible values of object identifiers are defined in a tree. The standards committees ISO and CCITT control the top of the tree and define portions of this tree. These standards committees delegate the remaining portions to other organizations so that each object class, attribute type, and attribute syntax has a unique object identifier. For example, the object identifier of the country object class is 2.5.6.2, which can also be written more verbosely as:

joint-iso-ccitt{2}modules{5}object classes{6}country{2}.

X.500 Naming Concepts

Information in the directory information base is organized in a hierarchical structure called a directory information tree. A hierarchical path, called a *distinguished name*, exists from the root of the tree to any entry in the directory information base. Each entry in the directory information base must have a name that uniquely describes that entry. For example, the employee (entry) David has the distinguished name C=US/O=hp/OU=hpind/CN=David, where C denotes the country, O the organization, OU the organization unit, and CN the common name.

The distinguished name is a collection of attribute type and attribute value pairs called *relative distinguished names*. From the example above, C (country) is an attribute type and US (United States) is an attribute value.

Alternative names are supported in the directory information tree through special entries called *alias entries*. An alias entry is a leaf entry in the directory information tree that points to another name. Alias entries do not contain any attributes other than their distinguished attributes because they have no subordinate entries.

GDS Components

As shown in Fig. 5, GDS is made up of four main components:

- Directory User Agent (DUA). This process is the user's representative to the directory service. The user can be a person or an application.
- Directory System Agent (DSA). This process controls and manages access to directory information.
- DUA Cache. This process keeps a cache of information obtained from the directory DSAs. One DUA cache runs on each client machine and is used by all the users on that machine. The DUA cache contains copies of recently accessed object entries and information about DSAs.
- Directory Information Base. This is where GDS stores information.

The DUA and DSA communicate by using the directory access protocol. DSAs use the directory system protocol to communicate with each other.

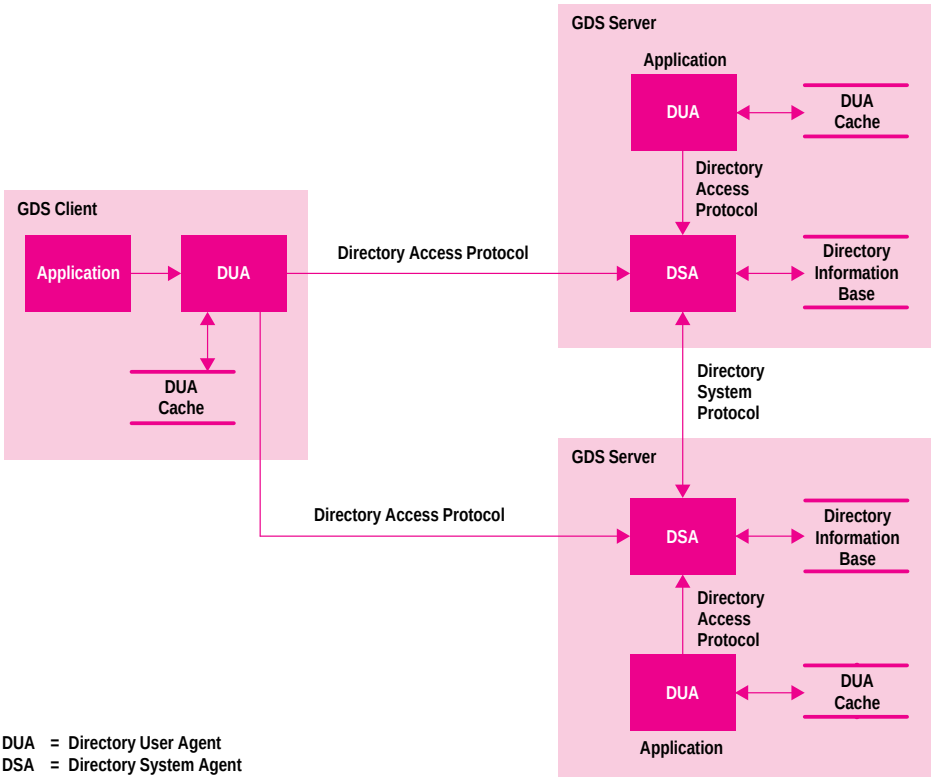

Fig. 5. Global Directory Service (GDS) components.

Since directory information is distributed over several DSAs, a DUA first directs any queries to a specific DSA. If this DSA does not have the information, there are two standard request methods that the DUA can use. The first method is referral—the DSA addressed returns the query to the DUA together with a reference indicating the other DSAs that have the information. Chaining is the second request method—the addressed DSA passes on the query directly to another DSA via the directory system protocol.

CDS Directory Structure

Every DCE cell must have at least one CDS server. The CDS servers in a cell collectively maintain a cell namespace, organized into a hierarchical directory structure. As mentioned above, the prefix /.: is shorthand for the global name of the cell and hence denotes the root of the cell namespace.

A CDS name is simply a series of directory names followed by an entry name. The directory names are ordered left-to-right from the cell root and are separated by the / character. For example, in the name /.:/acme/acme_server, the directory acme is a child of the cell root and the object acme_server is an entry in acme.

In a cell that contains more than one CDS server, CDS directories can be replicated, with each replica of a directory managed by a different CDS server. Among the replicas in a set, only one, the master replica, is modifiable; all other replicas are read-only. Replication increases the availability and reliability of CDS service and can ease recovery from hardware or network failure.

A CDS directory can contain three types of entries:

- Object entries contain information about objects in the cell. An object can be a host, a user, a server, a device, or any other resource that has a CDS name.
- Soft links provide alternate names for objects, directories, or other soft links.
- Child pointers are pointers to the directories contained by a parent directory. A child pointer is created when a new directory is created and is used by CDS to locate replicas of that directory when resolving the directory's name. Child pointers are created only by CDS itself, not by applications.

Like GDS, CDS stores information about named objects by associating attributes with names. Object entries might have attributes to store the object's UUID, its location, its access control list (ACL), the time it was created, and the time it was last updated. A soft link has an attribute to store the name of the object to which the link points.

Two special classes of CDS object entries warrant particular mention:

- RPC server entries store information about servers, including their location and the objects they manage, in the CDS database. Servers register this binding information, and clients look it up, via the RPC NSI interface.
- Junctions enable a service that implements its own namespace to splice that namespace into the DCE namespace. A junction is somewhat analogous to a mount point in a UNIX® file system; the junction entry stores binding information for a service and becomes the root for the namespace managed by that service. The CDS entry /.:/sec, for example, is the junction for the DCE security service. Applications can use names such as /.:/sec/principal/stravinsky to identify principals in the security registry and to obtain bindings to a security server. Similarly, /.:/fs is the junction for DFS, and /.:/hosts/<host-name>/config is the junction for the configuration services provided on each host by the DCE host daemon, dced.

CDS Components

CDS is a distributed service based on a client-server model. Fig. 6 illustrates the software components that implement this service.

All CDS directory data is stored in databases called *clearinghouses*, which are managed by CDS server daemons. The server daemon responds to requests from CDS clients for lookups in or updates to the database. When an RPC server invokes an RPC NSI API routine to export binding information to the namespace, for example, this routine triggers a CDS update operation. Similarly, when an RPC client imports bindings from the namespace, a CDS lookup operation is executed. Each CDS server keeps an image of its clearinghouse in memory and writes the clearinghouse periodically to disk.

A cell often includes more than one CDS server, each with its own clearinghouse. Running several CDS servers in one cell allows administrators to replicate CDS directories. If a directory is replicated, one clearinghouse stores the master replica of the directory, and other clearinghouses store read-only replicas. Clients can perform lookups from any replica but can perform updates only to the master replica. After a CDS entry is updated in the master replica of its directory, the CDS server that manages the master replica propagates the update to all CDS servers that manage read-only replicas. Replication improves responsiveness to clients by distributing work among several servers and ensures the availability of CDS service if a server machine fails or the network fails.

The architecture of CDS insulates applications from direct communication with CDS servers. To add, delete, or modify CDS data, applications call APIs such as the RPC NSI routines in the DCE run-time library. CDS client code in the library interacts with a daemon on the local host called the

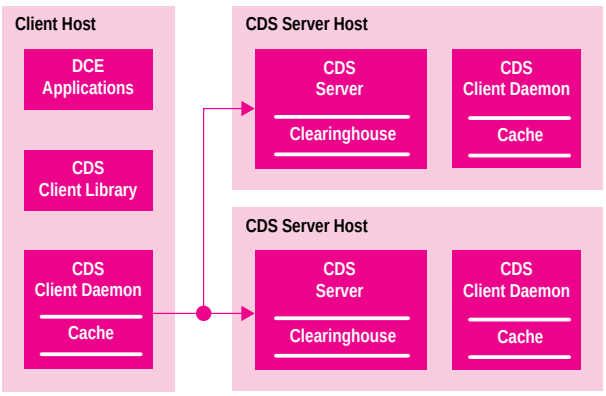

Fig. 6. Cell Directory Service (CDS) components.

CDS advertiser, which uses RPC to communicate as necessary with CDS servers. A CDS advertiser runs on every host in a DCE cell. (In many DCE implementations, several CDS client daemons execute on each host: one CDS advertiser and a number of CDS clerks. In the HP DCE/9000 product, a single CDS advertiser process subsumes all advertiser and clerk tasks.) To increase client performance, reduce server load, and reduce network traffic, each advertiser saves the results of its lookups in a cache. Frequently accessed data can be retrieved locally from the cache rather than via RPC from a server. The advertiser writes the cache to disk periodically, so cached data persists through reboots of the CDS client host.

Conclusion

DCE provides a three-level naming system and two naming APIs.

Names of cells are stored in a global namespace managed by a DNS server such as `named` or by an X.500 server such as the DCE Global Directory Service (GDS). The Global Directory Agent (GDA) oversees resolution of global cell names.

The cell namespace consists of two levels: the enterprise namespace managed by the DCE Cell Directory Service (CDS) and application namespaces such as those managed by DCE security and the DCE Distributed File Service (DFS). The roots of application namespaces are named by CDS junctions.

DCE offers two naming APIs. The RPC NSI interface is used by servers to register their names and locations in CDS and by clients to look up names and get back binding information. The XDS/XOM API can access names and their associated attributes in GDS and CDS.

The DCE name services have some limitations that X/Open's Federated Naming specification attempts to solve (see article, page 28). The RPC NSI API is a specialized interface that manages only RPC binding handle information; it cannot read or manipulate other attributes associated with a name. Many programmers find the XDS/XOM API cumbersome; this interface is also difficult to layer over other existing naming APIs. The RPC NSI API and the XDS/XOM API do not offer a way to create or delete directories programmatically, so an application that needs to create directories currently must use an internal CDS interface. The CDS and GDS protocols are complicated and not very general. New naming services that are introduced are unlikely to use either of these protocols or the XDS API. Finally, CDS does not support an easy, general way to create and resolve through junctions to application namespaces.

Acknowledgments

Liza Martin and Paul Smythe each reviewed several drafts of this article and made valuable suggestions, for which we are grateful.

References