Chapter 3

NDS Architecture

Overview

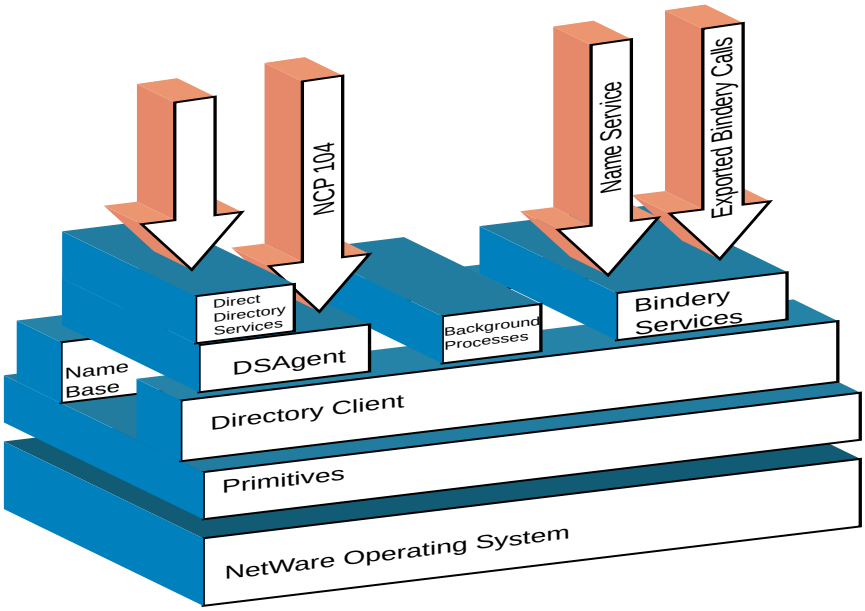
This chapter discusses the basic NetWare Directory Services architecture. Specifically, the chapter discusses:

- The NetWare Operating System.
- NDS architecture and subsystems.
- The NetWare Environment.
- NLM Design and Programming Issues.

NDS Architecture

NDS, because it is global, hierarchical, distributed, and replicated, is a complex database system. However, it has been designed as a set of interacting subsystems, as shown in the following illustration:

Figure 3.1
NDS Architecture



Having a basic understanding of these subsystems and what they do will give you a conceptual foundation that will help you understand the various functions of NDS as they are discussed in this volume. This chapter will briefly discuss each of these modules and how they interact with each other.

NDS Subsystems

This section describes the following NDS subsystems:

- Primitives
- Directory Client
- Name Base (Database Manager)
- DSAgent
- Direct Directory Services
- NCP 104
- Background Processes
- Bindery Services
- Exported Bindery Calls

Primitives

NetWare Directory Services (NDS) needs some services provided by the various operating systems to which it is ported. However, each operating system may provide these services differently.

Engineers use the *primitive* layer to provide a uniform view of operating system services, regardless of the operating system used. In other words, the primitives translate operating system functions into a form NDS can use. For example, the operating system may calculate time differently than NDS does. The primitive layer thus translates the operating system's time functions into a format useful to NDS. All NDS modules must communicate with the operating system through the primitives.

Currently, the primitive layer provides NDS with a uniform view of the following services:

- Encryption. This is used for authentication.
- Connection table. Every inbound packet has an ID in the connection table, which maps the ID to an NDS object.
- Debugging. This allows NDS to set values for displaying and printing according to the local operating system.
- Event handling. This allows NDS to notify external processes of events.
- File I/O. This allows NDS to communicate with the local operating system's file system.
- Global variables. This defines how global variables are used throughout the system.
- Memory. This provides for memory access and allows for debugging memory leaks.
- Name service interface. This provides an interface to the operating system's name service.
- NCP client. This provides an interface to the NCP client, allowing NDS to communicate using the server.
- NCP server. This allows NDS to register with the stack for NCPs it uses.
- SAP. This allows NDS to advertise itself using SAP. It also allows NDS to emulate the dynamic bindery and receive SAP information.
- Thread. This provides a thread model useful to NDS.
- Time. This converts the local operating system's time to allow NDS to lock and allocate events within seconds.

- Unicode. This loads the local operating system's Unicode table for translation.

In addition, the primitive layer includes general utilities that are not primitive interfaces to the operating system:

- Porting utilities. These utilities takes care of byte-swapping (such as High-Low formats) and alignment.
- Naming utilities. These utilities handles name parsing and conversion between the name formats NDS supports (such as full dot and slash formats).
- Miscellaneous utilities. These are general utilities that work with such things as Unicode strings, comparisons, and copies.

Directory Client

The Directory Client (DClient) allows an NDS server to act as a client and request services from another NDS server. This module is functionally equivalent to an X.500 Directory User Agent (DUA).

It provides an API for selected NDS verbs. The Directory Client includes the following functions, among others:

- Authentication routines. These allow the server to authenticate to other servers.
- List routines. These allow the server to list entries in a container it does not store.
- Read routines. These allow the server to read the attributes of objects it does not store.
- Resolve Name routines. These allow the server to resolve an object's name to its physical location in the network.
- Name routines. These provide for parsing names, changing name formats, and other functions.
- Transport routines. These include fragmentation routines and NCP subroutines that NDS uses.

Name Base (Database Manager)

The Name Base module provides the following functions:

- Provides routines to enforce the schema. The schema defines the relations objects and attributes can have in the NDS database.
- Contains the metaschema, or the rules defining the schema. For example, the metaschema defines two types of object, container and leaf objects.

- Provides access to the database (record manager).
- Generally implements the NCP verbs that NDS supports.
- Calculates Access Control Lists (ACLs).
- Provides the schema cache. The schema cache enforces the schema in the tree. Since the schema does not change very often, the schema cache contains precomputed object class and attribute definitions.
- Provides the entry cache. This cache stores the entry IDs of those entries whose attributes have been modified. Caching speeds synchronization, since NDS synchronizes only those objects whose entry IDs are listed in the entry cache.
- Provides an access point for the DSRepair utility to repair any problems with the database.

Name Base routines never call the Directory Client.

DSAgent

The Directory Services Agent (DSAgent) provides the server side of directory operations. It is functionally equivalent to an X.500 Directory System Agent (DSA). This subsystem contains a jump table for taking inbound NCP 104 requests from the wire and changes them to local format (agent routines). It checks rights and calls the appropriate routines in the Name Base or Directory Client.

If the DSAgent is closed, requesting clients and servers cannot access the local database. However, requesting entities can resolve names at another server.

Direct Directory Services

This subsystem provides functionality for local system calls, such as Install, into the local agent. It also performs local operations such as creating a volume. Because these local calls do not go out on the wire, they do not have to be fragmented into packets.

Background Processes

This subsystem comprises the NDS functions that occur automatically without user intervention. These functions include the following:

- Replica synchronization. This process propagates changes made to a replica to all replicas of a partition. Some changes attempt to synchronize within ten seconds, while less-important changes attempt to synchronize within five minutes by default.
- Schema synchronization. This process propagates changes made to the schema. The schema is organized as a Directory tree that is invisi-

ble to users. The schema tree is replicated on each server in the network. When changes are made to a replica of the schema tree, schema synchronization propagates these changes to the other replicas.

- Janitor. This process scans the database going backwards in a flat scan to find objects that need to be deleted or updated. For example, Janitor purges dynamic bindery objects created before the last time the bindery was opened. If it finds an external reference, it tries three times to verify that the reference applies to a real object. If it cannot find the object referenced, it deletes the external reference. If it finds a back link with OBT_MOVE obituary, it changes all affected external references.

- Purge. After an obituary has been sent to all replicas, signified by the message "all processed = yes," this process purges it, freeing memory. It then finds the oldest time stamp and sends it to Janitor, which cleans up everything with an earlier time stamp.

The Background Processes module has no entry point for external access. Background processes operate as they are scheduled and triggered by other NDS modules. Background processes can make calls to both the Directory Client and the Name Base, depending on whether the local server stores the information needed for the operation.

Bindery Services

This module of NetWare Directory Services provides backwards compatibility with Bindery-based NetWare servers and clients. This module contains the bindery context, which is a set of Entry IDs for containers at which the bindery context is set. Given the Entry ID of a bindery container and an entry's Distinguished Name in a request, the Bindery can provide access to the Name Base or Directory Client.

Generally the Bindery Services module makes only local calls to the Name Base. Occasionally, however, it makes calls to the Directory Client, such as when it needs to read a console operator.

For example, when a user authenticates to a server, the server wants to know if that user is a supervisor over it. A supervisor is a user who has management rights, or rights to grant rights, on a server's entry. When a supervisor authenticates to the server, a supervisor bit is set in the connection table, giving the supervisor all rights to the file system. If the server doesn't have a copy of its own object, it has to go to another server where its object does exist and check rights.

Exported Bindery Calls

Previous versions of the bindery exported some calls from the operating system. These calls included *ReadProperty* and *WriteProperty*, among

others. They are included in NDS for backwards compatibility to support older applications, such as CLIB, that rely on them.