Chapter 2

Theory of Operations

Overview

This chapter discusses Novell Directory Services' basic operating theory. It provides an overview of partitioning, replication, name resolution and synchronization.

You should read this chapter if you are not familiar with NDS's basic operations.

NDS Partitions and Replicas

Because the NDS Directory is hierarchical and object-oriented, it lends itself well to distribution, or partitioning. And because the database is partitioned, it can be efficiently replicated.

Partitions

The NDS Directory can be mapped as a tree structure with container entries forming subtrees. Each subtree can be either partitioned separately, or partitioned with parallel (sibling) container entries as long as they are all held in a common superior (root) container.

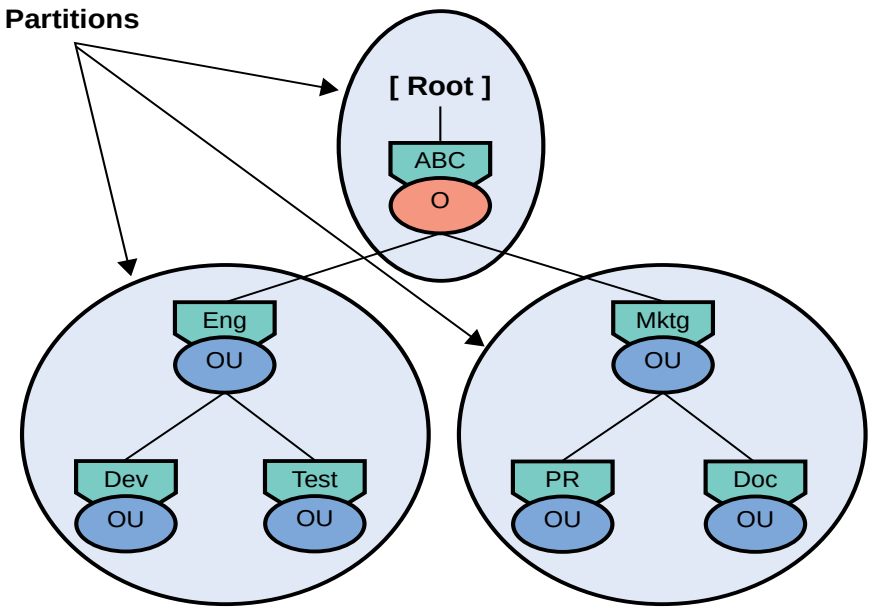
Partitioning Rules

Partitions obey the following rules:

- They must contain a connected subtree.
- They must contain only one container entry as the root of the subtree.
- They cannot overlap with any other part of the NDS tree.
- A partition is named by its root-most container entry (the container at the root of the subtree), which is called the *partition root.*

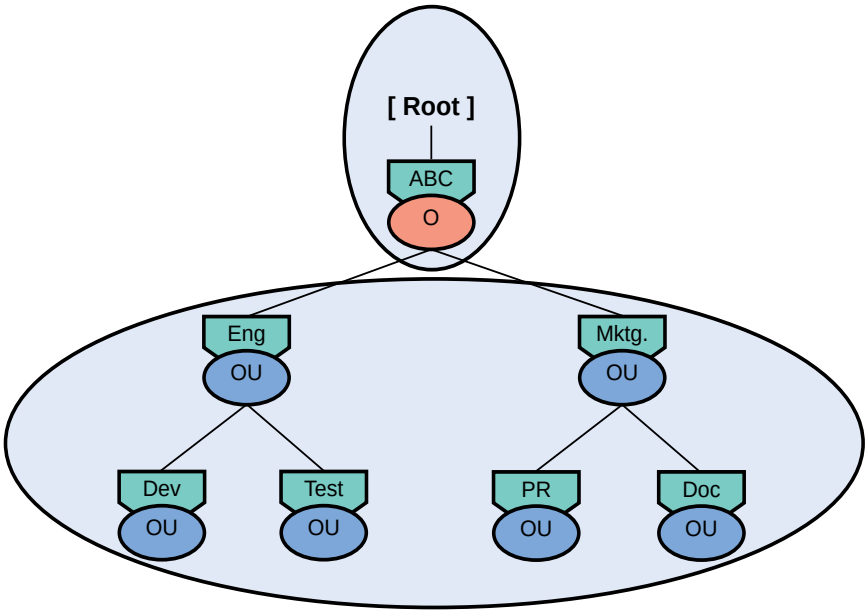
Figure 2.1 illustrates an example partitioned tree.

Figure 2.1
Example Partitioned Tree



The tree in Figure 2.1 has three partitions: [Root], Engineering and Marketing. Figure 2.2 shows an illegally partitioned tree in which Engineering and Marketing are combined as a subtree.

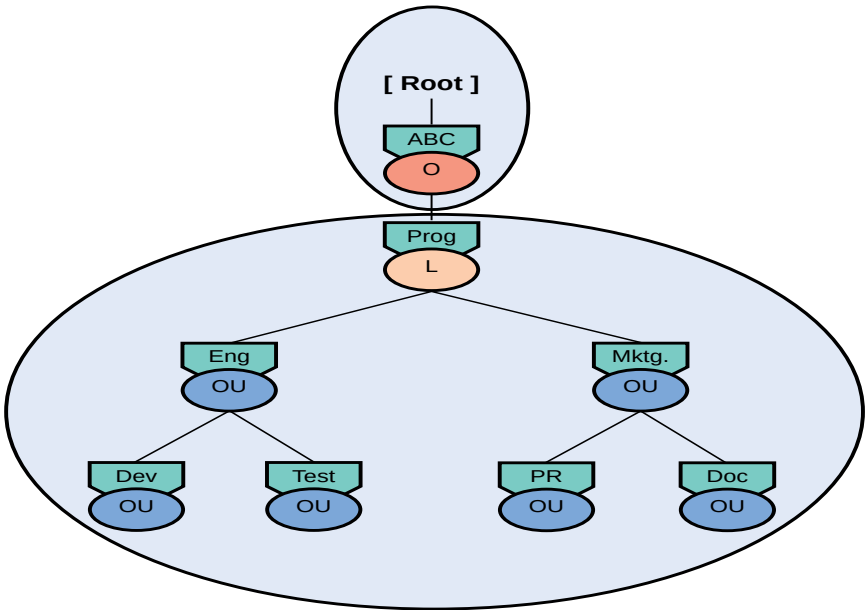
Figure 2.2
Improperly Partitioned Tree



As you can see in Figure 2.2, NDS has no way of accessing either Engineering or Marketing because the partition has combined subtrees. In other words, the partition cannot be named because it does not have a single, unique, root-most container (partition root).

However, as illustrated by Figure 2.3, if you join Engineering and Marketing into a single subtree (held in one container), you can then place them into a single partition.

Figure 2.3
Example Partitioned Tree with Combined Partitions

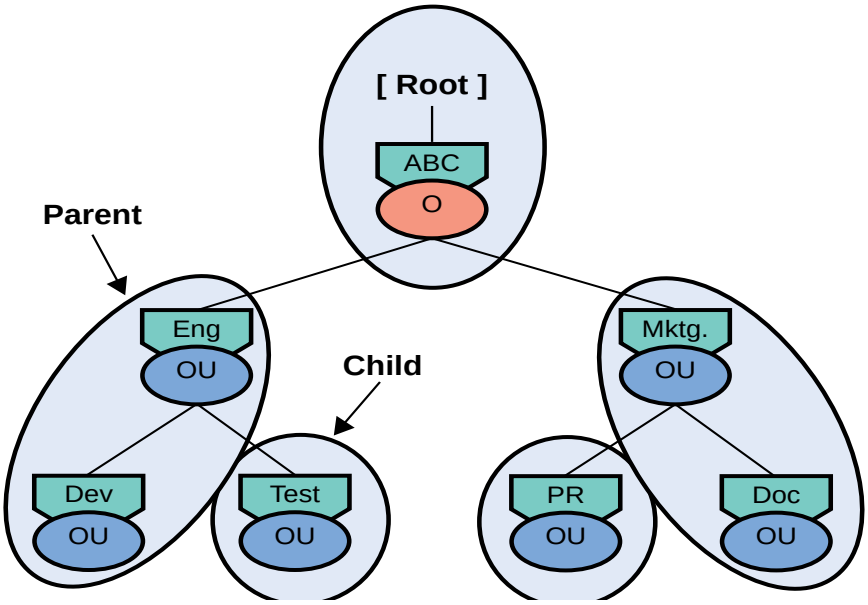


As Figure 2.3 shows, the partition is now named *Provo,* and the system has a way of locating it.

Parent and Child Partitions

When a container entry is put into a partition subordinate and independent of that entry's container, the partitions are known as *parent* and *child* partitions. The parent partition holds the superior container, and the child partition holds the subordinate container. This is illustrated in Figure 2.4

Figure 2.4
Parent and Child Partitions



Replicas

A single physical instance of a partition is called a replica. Partitions can have multiple replicas, but only one replica of a particular partition can exist on each server. (Keep in mind that servers can hold more than one replica, as long as each replica is of a different partition.)

Replica Types

Replicas must be designated as one of four types:

- Master
- Read/write
- Read-only
- Subordinate reference

One (and only one) replica of a partition must be designated as the master replica; the other replicas must be designated as either a read/write or read-only replica, or a subordinate reference. The replicas are invisible to the user; that is, the user does not know which replica contains the entries being accessed.

Master, Read/Write, and Read-Only Replicas. Clients can create, modify, and delete entries on either master or read/write replicas. However, clients can perform operations that deal with partitions only on the master replica. Clients cannot make any changes to read-only replicas.

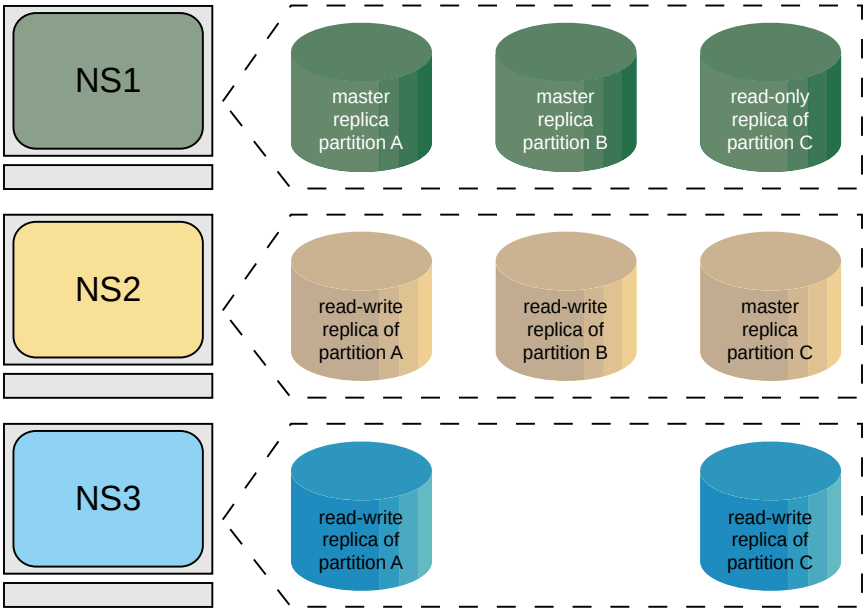
Important An entry must access either a master or a read/write replica in order to log in.s

Figure 2.5 shows three partitions (A, B, and C) replicated across three name servers (NS1, NS2, and NS3).

- NS1 stores the master replicas of partitions A and B and a read-only replica of partition C.
- NS2 stores the master replica of partition C and secondary replicas of A and B.
- NS3 stores secondary replicas of A and C.

Given this arrangement, any of the servers could handle a request to add an entry to partition A. Only NS1 and NS2 could handle a similar request for partition B, and only NS2 and NS3 could handle such a request for partition C.

Figure 2.5
Partition Replication



Only NS1 can create a new partition that is subordinate to partition A or B, and only NS2 can create a new partition that is subordinate to partition C.

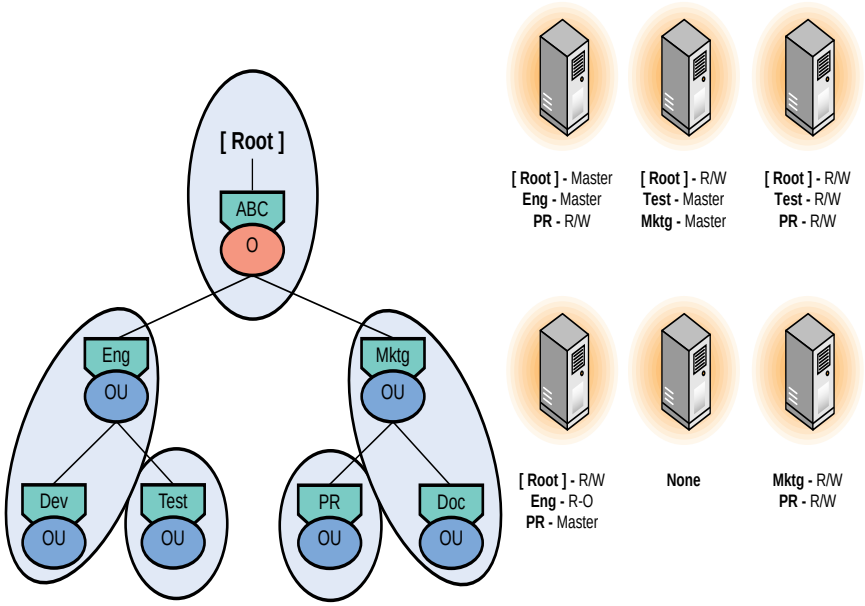
Subordinate References. Subordinate references, which are not visible to users, provide tree connectivity. Each subordinate reference is a complete copy of a given partition's root object, but is not a copy of the whole partition. As a general rule, subordinate references are placed on servers that contain a replica of a parent partition but not the relevant child partitions. In other words, a subordinate reference points to an absent subordinate partition. In this case, the server contains a subordinate reference for each child partition it does not store.

Subordinate references provide tree connectivity by referring to replicas

the server may need to find. Because the subordinate reference is a copy of a partition root object, it holds the Replica attribute, which lists all the servers on which replicas of the child partition can be found. NDS can then use this list to locate replicas of the subordinate partition.

Figure 2.6 illustrates a partitioned tree and its subsequent replica placement on the servers that are holding the tree.

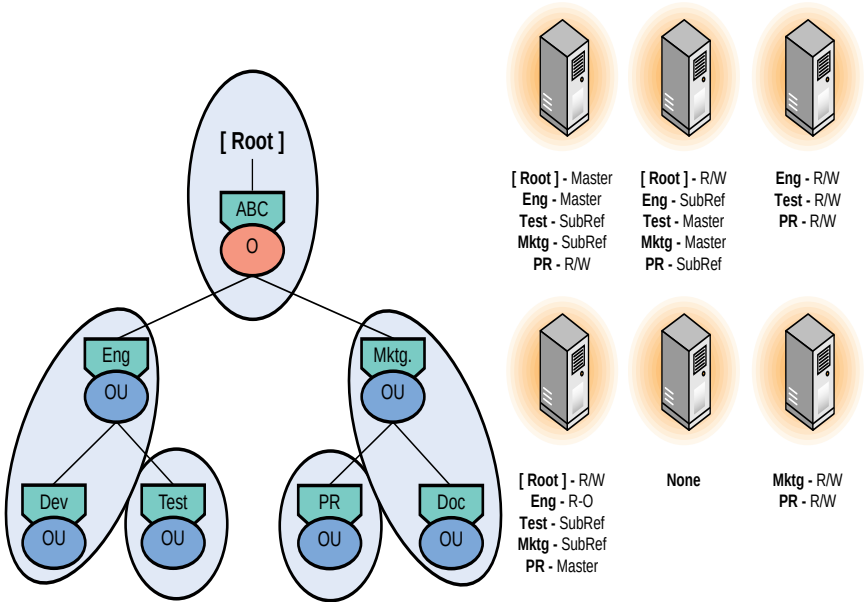
Figure 2.6
Replica Placement



.

Some of the servers in Figure 2.6 hold replicas of parent partitions but not replicas of the corresponding child. These servers must also hold subordinate references to the child partitions they do not hold, as shown in Figure 2.7. For example, because server Srv4 holds a replica of the Eng partition but not of Test, it must hold a subordinate reference to Test.

Figure 2.7
Replica Placement with Subordinate References



On server Srv1 in Figure 2.7, the subordinate reference of Mktg.ABC is a complete copy of the root object, Mktg.ABC, but not of its subordinate objects; the subordinate reference of Test.Eng.ABC is a complete copy of the entire partition, since Test.Eng.ABC is the only object in the partition. Users cannot change a subordinate reference's replica type.

Besides providing tree connectivity, subordinate references also help determine rights. Because a subordinate reference holds a copy of the partition root object, it holds that object's Inherited ACL attribute, which summarizes the Access Control Lists up to that point in the tree.

Replica List

Each replica contains a list of servers that support the partition it represents. The replica list is stored in each replica as a *Replica* attribute of the partition's root-most container entry. This list provides information needed for navigating the NDS tree and synchronizing the replicas. The replica list contains the following elements for each replica:

- *Server Name* The name of the server where the replica is located.
- *Replica Type* The type of the replica stored on the server designed in the Server Name field. (The type is either Master, R/W, RO, or SR.)
- *Replica State* The status of the replica. (The statuses include On, New, Replica dying, among others.)
- *Replica Number* The number that the master assigned to this rep-

lica at the time the replica was created.

- *Network Address* The server's address.
- *Remote ID* The Entry ID of the replica's partition root entry.

Name Resolution

To resolve an entry's name, NDS maps the name to the network address of a server where the entry is stored.

This operation requires NDS to locate a physical Entry ID on a specific server. An Entry ID is valid for an entry on a specific server. That is, it is local to each server and is not used as a global ID. Walking the tree involves finding the physical location of an entry and obtaining that entry's Entry ID. After obtaining an Entry ID, a client can get more information about the entry through a *Read* operation.

Valid Entry IDs are those below FF000000. The following IDs are reserved for special uses:

Table 2.1
Special-Use Entry IDs

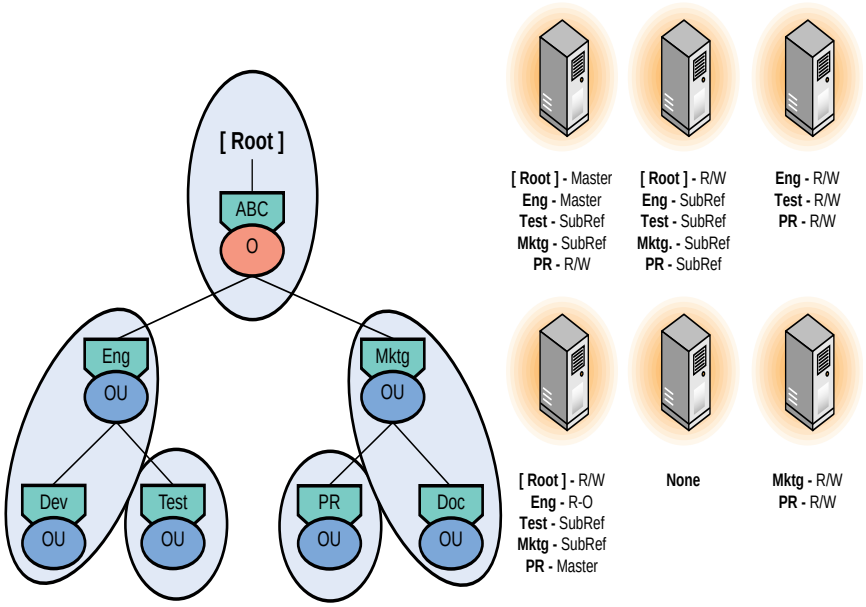
#define	Hex Representation	Description
ID_SUPERVISOR	0x01000000L	
ID_MIN_RESERVED	0xff000000L	
ID_PUBLIC	0xff000001L	ID for the [Public] entry
ID_INHERITANCE_-MASK	0xff000002L	ID for an Inherited Rights Filter
ID_CREATOR	0xff000003L	
ID_SELF	0xff000004L	
A_ENTRY_RIGHTS	0xff000005L	
A_SMS_RIGHTS	0xff000006L	
A_ALL_ATTRS_RIGHTS	0xff000007L	
A_REMOTE_ID_LIST	0xff000008L	
A_SCHEMAVALUE	0xff00000cL	
A_PASSWORD	0xff00000eL	
A_TREE_NAME	0xff000010L	
ID_ROOT_TEMPLATE	0xff000011L	
A_MONITORED_-CONNECTION	0xff000012L	
ID_DELETED_ENTRY	0xff000013L	ID for a deleted entry

NDS must resolve names in the following instances:

- Login
- Authentication
- Locating entries
- Reading entry information

For example, in the tree illustrated in Figure 2.8, a client connected to Srv3 must access a resource in the Test partition. In this case, Srv3 locally contains a replica of the Test partition, so it sends the requested information back to the client.

Figure 2.8
Resolving a Name



However, if the client is connected to server Srv4 and the same request is made, the server checks its local database and finds that it holds only a subordinate reference to Test. The client agent returns the replica list to the client, which shows where replicas of Test are stored. The client then determines which server is the least expensive to access and requests the information from that server.

If the client requests information from Srv5 (which has no replicas), the server sends referrals collected from SAP regarding the required resource. If the client requests information from Srv6, the client agent looks at the replica list and asks all the servers in the list for information regarding a more superior container in the tree.

Replica Synchronization

The NDS database is loosely consistent, which means that at any given instant any particular replica is not guaranteed to hold the latest changes to the database. However, to ensure the integrity of the database, NDS automatically synchronizes all the replicas, and the database is guaranteed to completely synchronize over any period of time during which no changes occur and when all servers remain connected. This synchronization process runs in the background.

The NDS synchronization process is divided into two types:

- Fast synchronization
- Slow synchronization

Fast synchronization occurs at ten-second intervals. All entry modification events are scheduled for fast synchronization.

Slow synchronization occurs at thirty-minute intervals. Only the attributes that deal with the login time and the login network address are scheduled for slow synchronization.

Although the X.500 standard suggests that all the information for every entry be exchanged during synchronization, NDS exchanges only the delta (or updated) information. Each attribute has a time stamp associated with it. NDS updates this time stamp whenever an entry's attribute is updated. A replica uses this time stamp and the time stamp associated with each replica in the replica list to determine whether the entry must be included in the current synchronization process.

NDS References

NDS relies on two types of references to help manage the database and provide tree connectivity:

- External references
- Back links

External References

Most operations involving an NDS object require the requesting entity to obtain an Entry ID associated with the object. The Entry ID is a 32-bit opaque value that serves as a handle and is valid indefinitely on a specific server.

If the server does not hold the requested Entry ID, it may create a temporary record of the entry in its Entry file. The temporary record is called an *external reference* and is valid for 8 days, when it is checked for validity.

External references contain all the attributes of the original entry with the following exceptions:

- The Class ID may be set to -1 if the external reference has not been backlinked.
- The Subordinate Count reflects only the subordinate entries held by the local server.
- The creation time may be set to 0.
- The modification time reflects the last time the external reference was accessed. If more than 8 days has passed since the last access, the Janitor process removes the external reference, unless needed for consistency.

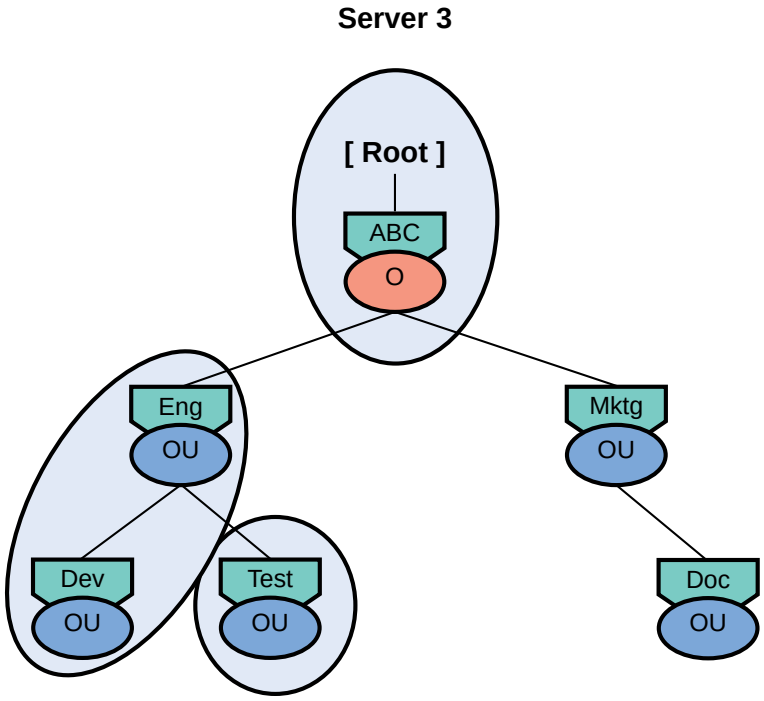
External references are created as placeholders for entries not stored on the local server. NDS tracks these nonlocal entries so that it can notify them of renamed entries, moved entries, and deleted entries. External references also provide full name resolution, complete tree hierarchy, and partial name matching. They have the following purposes:

- Act as placeholders for entries not physically located on the local server (external entries)
- Provide for authentication
- Provide tree connectivity
- Allow some attributes of the real entries to be cached in order to improve performance

Important An external reference is not a pointer; it is a placeholder for a real entry.

Referring to External Entries. An external reference refers to an entry that is not physically located on the local server. For example, in Figure 2.9, if a user located in the Test container on Srv3 must access a resource in the Documentation container, Srv3 would hold an external reference to Doc.Mktg.ABC.

Figure 2.9
External References



In this case, external references reduce the database size and allow a single point from which to rename and delete entries.

The following actions create external references for nonlocal entries:

- A nonlocal entry authenticates to or attaches to the server.
- A nonlocal entry is added as a file system or local entry trustee.

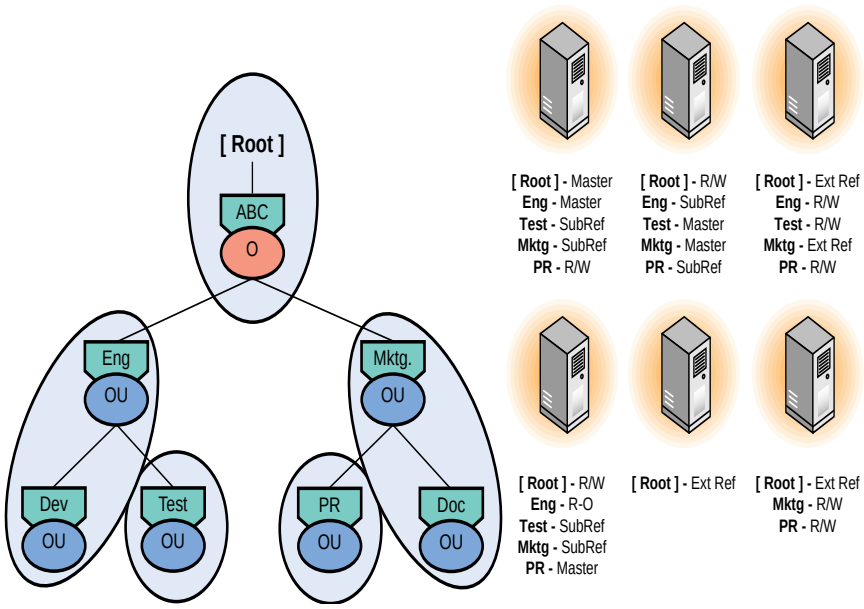
- A nonlocal entry is added as a member of a Group entry.

Providing for Authentication. External references also allow an object to authenticate to servers that do not hold copies of its object. Authentication requires an Entry ID for any object that is authenticating. For entries not stored on the local server, this Entry ID is supplied by an external reference. Because a server holding an entry must have reference to every entry in its Distinguished Name, the server will hold external references of the any of the entry's parent entries that it does not store, all the way up to the [Root] partition.

Also, as part of authentication NDS must calculate the security equivalence vector for the server's connection table. This vector includes all the entries that are security equivalent to the authenticating entry. NDS must create external references for any entries in the security equivalence vector that it does not store locally.

Providing Tree Connectivity. An external reference can also be thought of as supplying the missing link(s) between existing replicas on the server and the root. For example, in Figure 2.10, Srv5 does not physically hold a replica of [Root]. However, because Srv5 is part of the ABCTree, it must be able to resolve its local name to [Root]. Therefore, Srv5 would have an external reference to [Root]. Srv6 also does not have a replica of [Root], and it must also be able to link to the tree root. However, in this case Serv6 holds a replica of Mktg which is a subordinate of the ABC container. Therefore, Srv6 must hold an external reference to ABC and to [Root]. Figure 2.10 shows the external references required to connect the ABCTree.

Figure 2.10
Tree Connectivity



Caching Attributes of External Entries. External references allow frequently accessed attribute information to be cached by the server. This means that the server can access this information locally, instead of having to resort to another server. Currently, the following information is cached:

- The Public Key attribute is cached for all external references.
- For Server entries, the Status and DS Revision (build number) attributes are cached.

This cached information improves authentication performance. Cached external references are also referred to as *transient* external references. If an external reference is not accessed within a period of time (set by the network administrator), the server deletes the cached information. The public key is cached only during the current session. After the current session, the server deletes the cached information.

Back Links

When NDS creates a new external reference, it also attempts to create a pointer to the server holding the external reference as one of the attributes of the nonlocal entry. This pointer is called a back link and is stored as a Back Link attribute. If NDS is unable to create the back link, it continues trying to create the link 9 times. The default retry interval is currently 3 minutes. If NDS cannot create the back link after 9 times, the Back Link process creates the back link.

The Back Link process executes on a time interval set by the network

administrator. Currently, the default interval is 13 hours. The Back Link process has two basic functions:

- Remove any expired and non-required external references from the system.
- Create and maintain any back links not created at the same time as the external reference.

For information about the Back Link process, see Chapter 8, "Background Processes". r