CHAPTER 4

Overview of Netscape Directory Server

- ❑ Basic Installation
- ❑ A Brief Hands-on Tour of Netscape Directory Server
- ❑ Product Focus and Feature Set
- ❑ Extending the Netscape Server: A Simple Plug-in Example
- ❑ Further Reading
- ❑ Looking Ahead

A quick way to learn how to drive a car is to get behind the wheel, turn the key, and ease away from the curb out into traffic. Preferably, someone who is an experienced driver is seated in the passenger seat to coach you and help you master the rules of the road. Similarly, a good way to learn about directory software is to break open the shrink-wrap on a product, install it, and start turning the knobs and buttons to see what it can do.

This chapter provides a hands-on walk-through of one of the leading LDAP products, Netscape Directory Server. You will play the part of the beginning driver, and this chapter will act as the experienced coach sitting next to you. Our tour of version 6 of the Netscape server has four phases:

1. A walk through the basic installation instructions
2. A brief hands-on tour
3. A discussion of the focus and feature set
4. A demonstration of how to extend the server

Now it's time to get behind the wheel, turn the key, and hit the directory services road.

Basic Installation

First locate a system that meets Netscape's minimum requirements. Netscape Directory Server runs on several popular Unix platforms, including Sun Solaris, as well as on Microsoft Windows 2000 Server. Details of the specific system requirements can be found in the *Netscape Directory Server 6 Installation Guide*. This chapter provides detailed installation instructions for Solaris and Microsoft Windows 2000 Server. Table 4.1 summarizes the system requirements for both.

Once you have located a suitable system, place a copy of the Netscape Directory Server 6 installation package on that computer. For production use you must purchase the software, in which case you receive the software on CD-ROM from Netscape. A full-featured version can also be downloaded for evaluation purposes from the AOL Strategic Business Solutions Netscape Enterprise Web site at http://enterprise.netscape.com. The remainder of this section assumes that you have placed the installation package in the `/export` directory on a system running Solaris 8 or on a Windows 2000 system in the root of the `C:` drive.

A basic installation of Netscape Directory Server requires three steps:

Step 1: Extract and start the setup program.

Step 2: Answer a series of installation questions.

Step 3: Complete the installation and load data.

Table 4.1 System Requirements for Running Netscape Directory Server

	Requirement	
System Feature	**Solaris**	**Windows 2000 Server**
Operating system	Sun Solaris 8 with Sun's recommended patches	Windows 2000 Server or Advanced Server with Microsoft's latest service pack
Processor	UltraSPARC or better	Pentium II or better
Free disk space	200MB	200MB
Free RAM	256MB	256MB
Extraction utility	GNU zip (gzip)	Info-ZIP's UnZip, Nico Mak Computing's WinZip, or a similar utility to extract the contents of `.zip` files
Installation package filename for version 6.01	`directory-6.01-us.sparc-sun-solaris2.8.tar.gz`	`d601diu.zip`

To allow the directory server to accept LDAP connections on a TCP port below 1024 (such as the standard port, 389), you must execute the installation as the system superuser (root) on Solaris. On Windows 2000 you should perform the installation as a user that has administrator privileges.

Extracting and Starting the Setup Program

To extract and launch the setup program on Solaris, execute these commands:

```
su root
mkdir /export/dsinstall
cd /export/dsinstall
gzip -dc ../directory-6.01-us.sparc-sun-solaris2.8.tar.gz | tar -xvof -
./setup
```

To do the same on Windows 2000, execute these commands from the Windows command prompt:

```
md \dsinstall
cd dsinstall
unzip c:\d601diu.zip
setup
```

Figure 4.1 shows the first screen that is presented by the Netscape setup program on Solaris.

Answering Installation Questions

Netscape supports three installation modes:

1. **Express**. Minimal options; used for product evaluation only.
2. **Typical**. Recommended for most first-time installations.
3. **Custom**. For advanced installations.

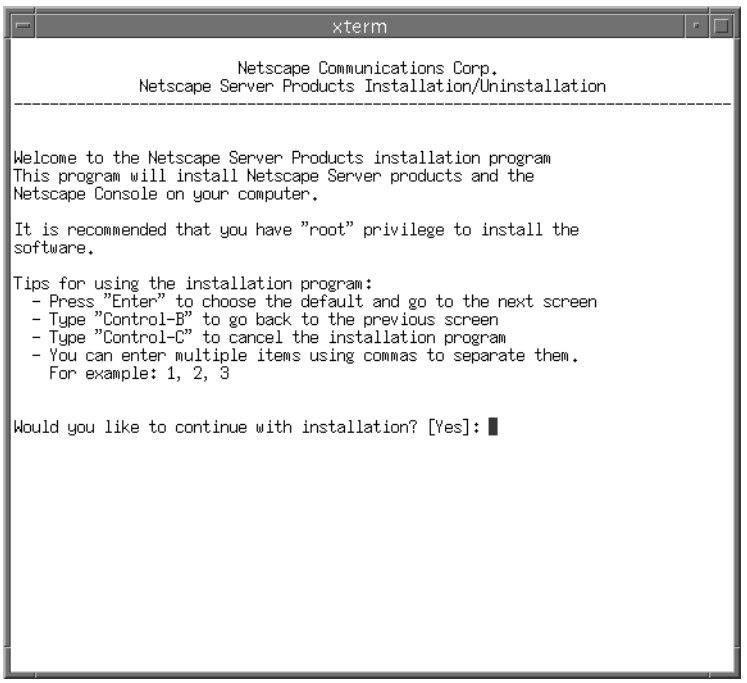
In this section the Typical mode is used, which is the default choice. The setup program presents a series of installation-related questions you must answer. On Solaris, follow these steps:

Step 1: Accept the default answers on each setup screen (except on the license screen, where you must type "Yes") until you see a prompt for "Install Location." Type "/export/ds6".

Step 2: Continue and accept the default answers on each setup screen until you see a prompt for "Directory Server Identifier." Type "example".

Step 3: On the next screen, which asks for an "Administrator ID," accept the default ID of "admin" and choose a password (the password is case sensitive). The adminis-

Figure 4.1 The First Netscape Directory Server Setup Screen on Solaris



trator identity is given full administrative rights to the configuration data in all directory servers.

Step 4: The next screen asks for your directory suffix; this is the base DN, or *naming context*, under which all of your directory's data resides (additional suffixes may be added later). Type "dc=example,dc=com" for the suffix.

Step 5: Accept the default directory manager DN on the next screen (`cn=Directory Manager`) and use the password "secret389" to ensure that the examples in the rest of this chapter work correctly.

Step 6: Accept the default answers for the remaining setup questions.

You are done when you reach a screen that says, "Extracting Netscape core components." Wait for the setup program to finish placing the directory server files on the disk.

On Microsoft Windows, follow these steps:

Step 1: Accept the default answers until you see a dialog box like the one shown in Figure 4.2 titled **Directory Server 6.0 Server Settings**. Type in "example" as the

Figure 4.2 The Directory Server Settings Dialog Box on Windows

server identifier, "389" as the server port, and "dc=example,dc=com" as the suffix (naming context).

Note

By default, Netscape Directory Server is configured to listen for incoming LDAP connections on TCP port 389, and the commands shown in this chapter assume port 389. If another server is already installed that is using port 389, disable or uninstall the other server (which is probably another LDAP server) before installing the Netscape server. If that is not possible, specify a different port in Netscape's Directory Server settings dialog during installation and remember what you chose. Then adjust the LDAP commands used later in this chapter as necessary to specify the port you chose (most commands use port 389 by default). For example, if you choose port 3389 when installing the server, you need to add `-p 3389` to the command-line parameters when issuing an `ldapsearch` or `ldapmodify` command.

Step 2: On the next dialog box, accept the default directory server administrator ID ("admin") and choose a password (the password is case sensitive).

Step 3: Accept the defaults on the remaining dialog boxes, except for the "Directory Server Manager" dialog box, where you should use a password of "secret389" to ensure that the examples in the rest of this chapter work correctly.

Step 4: When you arrive at the final **Configuration Summary** screen, double-check that everything looks correct, and press the **Enter** key or click the **Install** button.

Step 5: Wait for the setup program to finish placing the directory server files on the disk.

Completing the Installation and Loading Sample Data

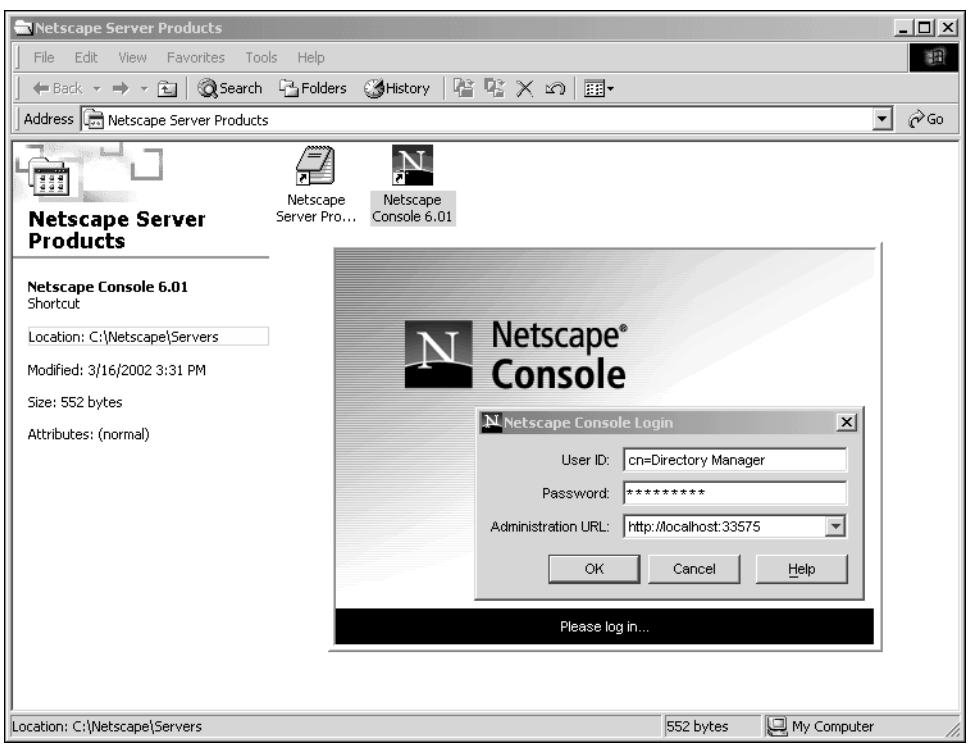
Once the files have been installed on the disk, the Netscape setup program automatically starts Directory Server as well as Administration Server, which is a specialized HTTP server. Netscape Directory Server can be configured and managed with a variety of command-line utilities or through use of a graphical point-and-click console interface named Netscape Console.

Step 1: Start Netscape Console by double-clicking on the **Netscape Console** icon on Microsoft Windows, or by typing these commands on Solaris:

```
cd /export/ds6
./startconsole
```

Netscape Console is a Java application, and it functions and looks the same on all platforms. Figure 4.3 shows the console login screen.

Figure 4.3 The Netscape Console Login Screen



Step 2: Log in with a user ID of "cn=Directory Manager" and a password of "secret389." Do not change the administration URL; it should be correct by default. After the main console window opens, expand the nodes within the **Servers and Applications** topology tree on the left side of the window until you see a node labeled **Directory Server (example)**. Double-click it. Figure 4.4 shows the Directory Server console window that opens.

Step 3: Load some sample data from the `Example.ldif` file that Netscape ships with its directory server. Click the **Import Databases** task button and type the path for the `Example.ldif` file. On Solaris, it is

```
/export/ds6/slapd-example/ldif/Example.ldif
```

On Microsoft Windows, the correct path is

```
C:\Netscape\Servers\slapd-example\ldif\Example.ldif
```

Figure 4.4 The Netscape Directory Server Console

Step 4: Click the **OK** button to import the data. You should see a message that reads "152 objects imported, 8 objects rejected." Ignore the rejected entries; the setup program created default entries with the same name as the eight rejected ones, and those entries will work for our purposes. The console import task does not overwrite existing data. After the data has been imported, use a text editor to look at the contents of the `Example.ldif` file. Listing 4.1 shows a few entries from `Example.ldif`.

Listing 4.1 A Few Entries from Netscape's `Example.ldif` File

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
aci: (target ="ldap:///dc=example,dc=com")(targetattr !=
 "userPassword")(version 3.0;acl "Anonymous read-search access";
 allow (read, search, compare)(userdn = "ldap:///anyone");)
aci: (target="ldap:///dc=example,dc=com") (targetattr =
  "*")(version 3.0; acl "allow all Admin group"; allow(all) groupdn =
  "ldap:///cn=Directory Administrators,ou=Groups,dc=example,dc=com";)

dn: ou=People, dc=example,dc=com
objectclass: top
objectclass: organizationalunit
ou: People
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr =
 "userpassword || telephonenumber || facsimiletelephonenumber")(version 3.0;
 acl "Allow self entry modification";allow (write)(userdn = "ldap:///self");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Accounting)")(version 3.0;
 acl "Accounting Managers Group Permissions";allow (write) (groupdn =
  "ldap:///cn=Accounting Managers,ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Human Resources)")(version 3.0;
 acl "HR Group Permissions";allow (write)(groupdn = "ldap:///cn=HR Managers,
 ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn ||sn || uid")(targetfilter ="(ou=Product Testing)")(version 3.0;
 acl "QA Group Permissions";allow (write)(groupdn = "ldap:///cn=QA Managers,
 ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !=
 "cn || sn || uid")(targetfilter ="(ou=Product Development)")(version 3.0;
 acl "Engineering Group Permissions";allow (write)(groupdn = "ldap:///
 cn=PD Managers,ou=groups,dc=example,dc=com");)

dn: uid=bjensen, ou=People, dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
sn: Jensen
givenname: Barbara
objectclass: top
objectclass: person
```

```
objectclass: organizationalPerson
objectclass: inetOrgPerson
ou: Product Development
ou: People
L: Cupertino
uid: bjensen
mail: bjensen@example.com
telephonenumber: +1 408 555 1862
facsimiletelephonenumber: +1 408 555 1992
roomnumber: 0209
userpassword: hifalutin
```

The `aci` attributes hold Netscape-specific access control information. The access control features of Netscape Directory Server are discussed later in this chapter.

Finally, let's confirm that the sample data has been loaded.

Step 5: Click the **Directory** tab near the top of the **Netscape Console** window to see a tree view of the directory information tree (DIT). Click to expand the node labeled **example** (which is a `domain` entry) and select the **People** container (an `organizationalUnit` entry) by clicking on it. A list of user IDs will appear in the right-hand side of the window. The list contains the relative distinguished names (RDNs) of all the entries that are children of the `ou=People,dc=example,dc=com` entry. Double-click any ID to see the attributes of that person. Figure 4.5 shows `bjensen`'s (Barbara Jensen's) entry.

Step 6: To see all of the LDAP attributes and values in tabular form, click the **Advanced...** button.

Congratulations! You have managed to find first gear, pull away from the curb, and start the car moving down the street.

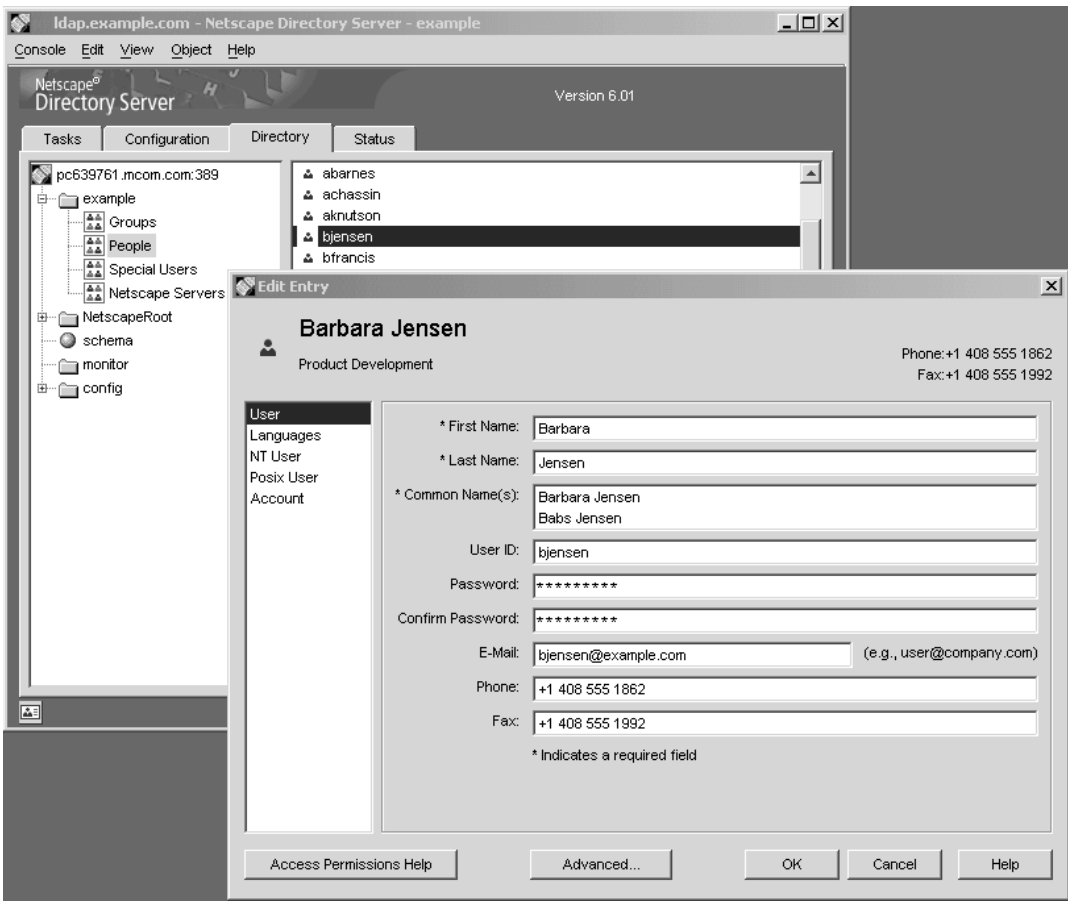
A Brief Hands-on Tour of Netscape Directory Server

Now that you have a functioning LDAP server, you can explore its capabilities. Put another way, it is time to shift into second gear. The Netscape server supports all significant LDAP standards, including LDAPv2 (for compatibility with very old applications) and LDAPv3. Netscape also supports many proposed LDAPv3 extensions, which are discussed later in the chapter.

Searching

When you viewed the `dc=example,dc=com` entries after installing the server, you used the **Directory** tab within the Netscape Directory Server console. Now add a new entry using that same interface.

Figure 4.5 Viewing the Barbara Jensen Sample Entry

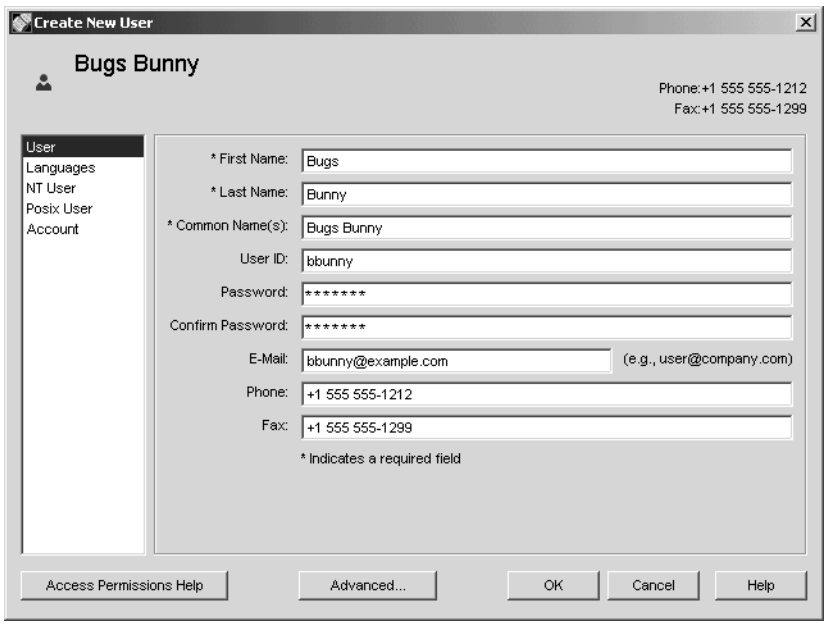


Step 1: Return to the **Directory** tab and select the **People** node on the left side of the window. Within the **Object** menu, execute the **New User** command to open the **Create New User** window. Using the information shown in Figure 4.6, create a new user named Bugs Bunny. Choose a password such as "@home@WB" (the password is not used in any of the examples in this chapter).

Next let's execute some directory searches.

Step 2: Use Netscape Console to search for the entry you just added. Locate the entry labeled **People** on the left side of the **Directory** pane. Use the rightmost mouse button to click on the **People** entry and select **Search...** from the context

Figure 4.6 Creating a New Entry for Bugs Bunny



menu that appears. Type "Bugs Bunny" in the text field labeled **for** and press the **Enter** key.

A search result list with one entry should appear. Feel free to try other searches as well. The Netscape Console search window supports several search modes, including one that allows you to type arbitrary LDAP filters.

Step 3: Search for the same entry using the ldapsearch command-line tool that is bundled with the Netscape server. Start a Unix shell or a Microsoft Windows command prompt window. On Solaris, type these commands:

```
cd /export/ds6/shared/bin
./ldapsearch -b "dc=example,dc=com" "(cn=Bugs Bunny)"
```

On Microsoft Windows, type these commands:

```
cd \Netscape\Servers\shared\bin
ldapsearch -b "dc=example,dc=com" "(cn=Bugs Bunny)"
```

In the `ldapsearch` command the argument to the `-b` option is the search base (`dc=example,dc=com`), and the last command-line parameter is the LDAP filter `(cn=Bugs Bunny)`, which specifies an exact match on the `cn` attribute for the string "Bugs Bunny." Listing 4.2 shows the result of this search: one entry in LDIF format.

Listing 4.2 Result of Search for "Bugs Bunny"

```
version: 1
dn: uid=bbunny,ou=People, dc=example,dc=com
mail: bbunny@example.com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetorgperson
givenName: Bugs
telephoneNumber: +1 555 555 1212
cn: Bugs Bunny
uid: bbunny
sn: Bunny
facsimileTelephoneNumber: +1 555 555 1299
```

There are no surprises here. Netscape has standardized on the `inetOrgPerson` object class defined in RFC 2798 for user entries, so the entry has `objectClass: inetorgperson`. The entry's RDN is `uid=bbunny` because by default Netscape Console uses the `uid` (user id) attribute to name user entries.

Step 4: Try a more complex search. Suppose you want to find all entries that are people in the Product Development department who are located in the Cupertino location. Further, suppose that you want to retrieve only the name, department, and e-mail address of each person. This `ldapsearch` command will do the job (execute this—and all `ldapsearch` commands shown in this chapter—from the same directory as in step 3):

```
ldapsearch -b "dc=example,dc=com"
 ↳"(&(ou=Product Development)(L=Cupertino))" cn ou mail
```

The search filter consists of two equality filters ANDed together, and the list of requested attributes appears at the end of the command line (`cn ou mail`). This search returns 11 entries. Wouldn't it be nice if they were sorted alphabetically by name? Luckily, ldapsearch supports the LDAPv3 Server-Side Sorting control, and so does Netscape Directory Server.

Step 5: Add the `-x` option to tell the server to sort the entries before returning them, and add the `-Scn` option to specify that the `cn` (common name) attribute should be used as the sort key. Here is the revised `ldapsearch` command:

```
ldapsearch -b "dc=example,dc=com" -x -Scn
 ↳"(&(ou=Product Development)(L=Cupertino))" cn ou mail
```

Listing 4.3 shows the resulting LDIF output.

Listing 4.3 Search Results Sorted by Name

```
version: 1
dn: uid=aworrell, ou=People, dc=example,dc=com
cn: Alan Worrell
ou: Product Development
ou: People
mail: aworrell@example.com

dn: uid=aknutson, ou=People, dc=example,dc=com
cn: Ashley Knutson
ou: Product Development
ou: People
mail: aknutson@example.com

dn: uid=bjensen, ou=People, dc=example,dc=com
cn: Barbara Jensen
cn: Babs Jensen
ou: Product Development
ou: People
mail: bjensen@example.com

dn: uid=cwallace, ou=People, dc=example,dc=com
cn: Cecil Wallace
ou: Product Development
ou: People
mail: cwallace@example.com

dn: uid=jmuffly, ou=People, dc=example,dc=com
cn: Jeff Muffly
ou: Product Development
ou: People
mail: jmuffly@example.com

dn: uid=jcampaig, ou=People, dc=example,dc=com
cn: Jody Campaigne
ou: Product Development
ou: People
mail: jcampaig@example.com

dn: uid=jbourke, ou=People, dc=example,dc=com
cn: Jon Bourke
ou: Product Development
ou: People
mail: jbourke@example.com

dn: uid=mlangdon, ou=People, dc=example,dc=com
cn: Marcus Langdon
ou: Product Development
ou: People
mail: mlangdon@example.com
```

```
dn: uid=mtalbot, ou=People, dc=example,dc=com
cn: Martin Talbot
ou: Product Development
ou: People
mail: mtalbot@example.com

dn: uid=smason, ou=People, dc=example,dc=com
cn: Sue Mason
ou: Product Development
ou: People
mail: smason@example.com

dn: uid=speterso, ou=People, dc=example,dc=com
cn: Sue Peterson
ou: Product Development
ou: People
mail: speterso@example.com
```

Manipulating Netscape Directory Server Databases

Netscape Directory Server uses a high-performance embedded database to store data, and it allows multiple database instances to be active at the same time. Each database instance has a unique name and stores data for one naming context (one subtree within the DIT). The Typical installation used earlier in the chapter created two database instances:

1. **`NetscapeRoot`**. Holds configuration and administration information that may be shared by more than one Netscape server.
2. **`userRoot`**. Holds the data that you load into the directory. Earlier in the chapter, you chose `dc=example,dc=com` as the naming context for your data.

Netscape Console has full support for creating and maintaining databases. This section shows how to manipulate directory databases from the command line.

Step 1: Before executing any of the commands shown, ensure that the current working directory is the `slapd-example` instance directory. Start a Unix shell or a Microsoft Windows command prompt window. On Solaris, type this command:

```
cd /export/ds6/slapd-example
```

On Microsoft Windows, type this command:

```
cd \Netscape\Servers\slapd-example
```

In the commands that follow, all of the leading "`./`" sequences should be omitted if you're working on Microsoft Windows.

Step 2: First use the `suffix2instance` command to display a list of active suffixes (naming contexts) and their corresponding Netscape Directory Server databases:

```
./suffix2instance -s ""
```

The `-s ""` parameter says that all suffixes are to be listed; if you wanted to list suffixes under only a specific subtree, you would include the subtree DN after the `-s`. The output produced is

```
Suffix, Instance name pair(s) under "":
        suffix "o=NetscapeRoot"; instance name "NetscapeRoot"
        suffix "dc=example,dc=com"; instance name "userRoot"
```

Step 3: Shut down the server using the `stop-slapd` command and replace the contents of the `userRoot` database with new data, a process known as *bulk-loading*. Listing 4.4 shows how to use Netscape's `ldif2db` command to load the `Example.ldif` file that is bundled with the server. The commands you need to type are shown in bold.

Listing 4.4 Bulk-Loading of `Example.ldif` Using the `ldif2db` Command

```
./stop-slapd
./ldif2db -n userRoot -i - <ldif/Example.ldif
importing data ...
[26/Aug/2002:16:04:18 -0500] - import userRoot: Index buffering enabled
↳with bucket size 15
[26/Aug/2002:16:04:18 -0500] - import userRoot: Beginning import job...
[26/Aug/2002:16:04:18 -0500] - import userRoot: Processing file stdin
[26/Aug/2002:16:04:19 -0500] - import userRoot: Finished scanning file stdin
↳  (160 entries)
[26/Aug/2002:16:04:19 -0500] - import userRoot: Workers finished; cleaning
↳  up...
[26/Aug/2002:16:04:22 -0500] - import userRoot: Workers cleaned up.
[26/Aug/2002:16:04:22 -0500] - import userRoot: Cleaning up producer
↳  thread...
[26/Aug/2002:16:04:22 -0500] - import userRoot: Indexing complete.
↳  Post-processing...
[26/Aug/2002:16:04:22 -0500] - import userRoot: Flushing caches...
[26/Aug/2002:16:04:22 -0500] - import userRoot: Closing files...
[26/Aug/2002:16:04:22 -0500] - import userRoot: Import complete.
↳  Processed 160 entries in 4 seconds. (40.00 entries/sec)
```

The `-n userRoot` parameter selects the `userRoot` database instance, `-i -` indicates that the LDIF file is provided on standard input, and `<ldif/Example.ldif` causes the shell to send the contents of the `Example.ldif` file to `ldif2db`'s standard input.

Listing 4.5 demonstrates the reverse process (creating an LDIF file from an existing database). On Windows, ensure that a directory named `\tmp` exists in the root of the drive where you installed Netscape Directory Server, or use a different pathname for the `example-dump.ldif` output file.

Listing 4.5 Dumping a Database Using the `db2ldif` Command

```
./db2ldif -n userRoot -a /tmp/example-dump.ldif
ldiffile: /tmp/example-dump.ldif
[26/Aug/2002:16:35:28 -0500] - export userRoot: Processed 160 entries (100%).
more < /tmp/example-dump.ldif
version: 1

# entry-id: 1
dn: dc=example,dc=com
objectClass: top
objectClass: domain
dc: example
aci: (target ="ldap:///dc=example,dc=com")(targetattr
!="userPassword")(versio
 n 3.0;acl "Anonymous read-search access";allow (read, search, compare)(userd
 n = "ldap:///anyone");)
aci: (target="ldap:///dc=example,dc=com") (targetattr = "*")(version 3.0; acl
 "allow all Admin group"; allow(all) groupdn = "ldap:///cn=Directory Administ
 rators,ou=Groups,dc=example,dc=com";)
nsUniqueId: 093e751b-1dd211b2-80000000-00000000

# entry-id: 2
dn: ou=Groups, dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: Groups
nsUniqueId: 093e751c-1dd211b2-80000000-00000000

# entry-id: 3
dn: cn=Directory Administrators, ou=Groups, dc=example,dc=com
cn: Directory Administrators
objectClass: top
objectClass: groupofuniquenames
ou: Groups
uniqueMember: uid=kvaughan, ou=People, dc=example,dc=com
uniqueMember: uid=rdaugherty, ou=People, dc=example,dc=com
uniqueMember: uid=hmiller, ou=People, dc=example,dc=com
nsUniqueId: 093e751d-1dd211b2-80000000-00000000
--More--(2%)
```

The `-n userRoot` parameter indicates that entries within the default directory database should be extracted (`-s "dc=example,dc=com"` may be used instead to extract the entries on the basis of the LDAP subtree that contains them). The `-a -` parameter says that the output should be sent to standard output, and `>example-dump.ldif` tells the shell to capture the output in a file named `example-dump.ldif`.

Listing 4.5 shows the first portion of the file, as viewed using the `more` command. Each entry in the LDIF file includes an `nsUniqueID` attribute, which is an operational attribute that holds a global unique identifier (GUID) generated by Netscape

Directory Server. The `nsUniqueID` values are used internally by Netscape to support replication and are preserved when entries are renamed. These values may also be used by LDAP clients that need to track entries without relying on the entry's DN.

Listing 4.6 shows how to dump a Netscape Directory Server database in Directory Services Markup Language (DSML) format rather than LDIF format. DSML is an XML-based format for representing directory data.

Listing 4.6 Dumping a Database Using the `db2dsml` Command

```
./db2dsml -n userRoot -a /tmp/example-dump.dsml
ldiffile: -
[26/Aug/2002:16:41:12 -0500] - export userRoot: Processed 160 entries (100%).
more < /tmp/example-dump.dsml
<?xml version="1.0" encoding="UTF-8" ?>
<dsml:dsml xmlns:dsml="http://www.dsml.org/DSML">
  <dsml:directory-entries>
  <dsml:entry dn="dc=example,dc=com">
    <dsml:objectclass>
      <dsml:oc-value>top</dsml:oc-value>
      <dsml:oc-value>domain</dsml:oc-value>
    </dsml:objectclass>
    <dsml:attr name="dc">
      <dsml:value>example</dsml:value>
    </dsml:attr>
    <dsml:attr name="nsuniqueid">
      <dsml:value>093e751b-1dd211b2-80000000-00000000</dsml:value>
    </dsml:attr>
    <dsml:attr name="aci">
      <dsml:value>(target ="ldap:///dc=example,dc=com")(targetattr !=
↳   "userPassword")(version 3.0;acl "Anonymous read-search access"
↳   ;allow (read, search, compare)(userdn = "ldap:///anyone");)</dsml:value>
      <dsml:value>(target="ldap:///dc=example,dc=com") (targetattr =
↳   "*")(version 3.0; acl "allow all Admin group"; allow(all)
↳   groupdn = "ldap:///cn=Directory Administrators,ou=Groups,
↳   dc=example,dc=com";)</dsml:value>
    </dsml:attr>
  </dsml:entry>
  <dsml:entry dn="ou=Groups, dc=example,dc=com">
    <dsml:objectclass>
      <dsml:oc-value>top</dsml:oc-value>
      <dsml:oc-value>organizationalunit</dsml:oc-value>
    </dsml:objectclass>
    <dsml:attr name="nsuniqueid">
      <dsml:value>093e751c-1dd211b2-80000000-00000000</dsml:value>
    </dsml:attr>
    <dsml:attr name="ou">
      <dsml:value>Groups</dsml:value>
    </dsml:attr>
  </dsml:entry>
  <dsml:entry dn="cn=Directory Administrators, ou=Groups, dc=example,dc=com">
```

```
    <dsml:objectclass>
      <dsml:oc-value>top</dsml:oc-value>
      <dsml:oc-value>groupofuniquenames</dsml:oc-value>
    </dsml:objectclass>
    <dsml:attr name="cn">
      <dsml:value>Directory Administrators</dsml:value>
    </dsml:attr>
    <dsml:attr name="nsuniqueid">
      <dsml:value>093e751d-1dd211b2-80000000-00000000</dsml:value>
    </dsml:attr>
    <dsml:attr name="uniquemember">
      <dsml:value>uid=kvaughan, ou=People, dc=example,dc=com</dsml:value>
      <dsml:value>uid=rdaugherty, ou=People, dc=example,dc=com</dsml:value>
      <dsml:value>uid=hmiller, ou=People, dc=example,dc=com</dsml:value>
    </dsml:attr>
    <dsml:attr name="ou">
      <dsml:value>Groups</dsml:value>
    </dsml:attr>
  </dsml:entry>
--More--(1%)
```

The DSML output is more verbose than LDIF, but it is useful if you want to work with XML-savvy tools and with other applications that understand XML. See Chapter 3, LDAPv3 Extensions, for more information about DSML.

Another important database maintenance task is creating backups of the data. Netscape supports *hot backups*—that is, backups that are performed while the directory server is running and accepting updates. The Netscape server stores its active database files under a subdirectory named `db`, and the hot backup process makes a transactionally consistent copy of all the files.

Step 1: While the server is running, use the `db2bak` command as shown in Listing 4.7 to create a complete backup of the directory data. The sample run is from a Solaris system. Each `.db3` file stores some entry data or an attribute index; on Microsoft Windows the pathnames of the files will be different.

Listing 4.7 Starting the Server and Creating a Backup

```
./start-slapd
./db2bak
[26/Aug/2002:17:01:21 -0500] - Backing up file 1 (/export/ds6/slapd-example/
↳ bak/2002_08_26_17_01_20/userRoot/id2entry.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 2 (/export/ds6/slapd-example/
↳  bak/2002_08_26_17_01_20/userRoot/entrydn.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 3 (/export/ds6/slapd-example/
↳  bak/2002_08_26_17_01_20/userRoot/parentid.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 4 (/export/ds6/slapd-example/
↳  bak/2002_08_26_17_01_20/userRoot/aci.db3)
```

```
[26/Aug/2002:17:01:21 -0500] - Backing up file 5 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/userRoot/uid.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 6 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/userRoot/nsUniqueId.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 7 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/userRoot/objectclass.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 8 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/userRoot/mail.db3)
[26/Aug/2002:17:01:21 -0500] - Backing up file 9 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/userRoot/cn.db3)
...
[26/Aug/2002:17:01:22 -0500] - Backing up file 25 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/NetscapeRoot/sn.db3)
[26/Aug/2002:17:01:22 -0500] - Backing up file 26 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/NetscapeRoot/givenName.db3)
[26/Aug/2002:17:01:22 -0500] - Backing up file 27 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/NetscapeRoot/uid.db3)
[26/Aug/2002:17:01:22 -0500] - Backing up file 28 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/NetscapeRoot/uniquemember.db3)
[26/Aug/2002:17:01:22 -0500] - Backing up file 29 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/log.0000000001)
[26/Aug/2002:17:01:22 -0500] - Backing up file 30 (/export/ds6/slapd-example/
↳   bak/2002_08_26_17_01_20/DBVERSION)
```

The `db2bak` command creates a consistent copy of all the database files. By default, the files are stored under a directory in the file system named according to the current date and time (`/export/ds6/slapd-example/bak/2002_08_26_17_01_20/` in the sample run shown).

Step 2: If running on Microsoft Windows, use the commands shown in Listing 4.8 to restore the directory server data from the backup you created. First, stop the server using the `stop-slapd` command (databases cannot be restored while the server is running). Next, simulate loss of the active database files by using the `del` or `rm` command to remove all of the `.db3` files. Finally, execute the `bak2db` command to restore the database files from the backup.

Listing 4.8 Restoring a Database from a Backup

```
stop-slapd
del /S/Q db\*.db3
Deleted file - C:\Netscape\Servers\slapd-example\db\DBVERSION
Deleted file - C:\Netscape\Servers\slapd-example\db\log.0000000001
Deleted file - C:\Netscape\Servers\slapd-example\db\NetscapeRoot\aci.db3
Deleted file - C:\Netscape\Servers\slapd-
↳   example\db\NetscapeRoot\ancestorid.db3
Deleted file - C:\Netscape\Servers\slapd-example\db\NetscapeRoot\cn.db3
...
Deleted file - C:\Netscape\Servers\slapd-example\db\userRoot\parentid.db3
Deleted file - C:\Netscape\Servers\slapd-example\db\userRoot\sn.db3
```

```
Deleted file - C:\Netscape\Servers\slapd-example\db\userRoot\telephoneNumber.db3
Deleted file - C:\Netscape\Servers\slapd-example\db\userRoot\uid.db3
Deleted file - C:\Netscape\Servers\slapd-example\db\userRoot\uniquemember.db3

bak2db C:\Netscape\Servers\slapd-example\bak\2002_08_26_170120
 [27/Aug/2002:10:06:40 -0500] - Restoring file 1
↳   (C:/Netscape/Servers/slapd-example/db/DBVERSION)
[27/Aug/2002:10:06:40 -0500] - Restoring file 2
↳   (C:/Netscape/Servers/slapd-example/db/log.0000000001)
[27/Aug/2002:10:06:43 -0500] - Restoring file 3
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/aci.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 4
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/ancestorid.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 5
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/cn.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 6
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/entrydn.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 7
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/givenName.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 8
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/id2entry.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 9
↳   (C:/Netscape/Servers/slapd-example/db/NetscapeRoot/nsUniqueId.db3)
...
[27/Aug/2002:10:06:43 -0500] - Restoring file 25
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/objectclass.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 26
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/parentid.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 27
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/sn.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 28
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/telephoneNumber.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 29
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/uid.db3)
[27/Aug/2002:10:06:43 -0500] - Restoring file 30
↳   (C:/Netscape/Servers/slapd-example/db/userRoot/uniquemember.db3)
```

If running on Solaris, use the following commands instead of the ones shown in Listing 4.8:

```
./stop-slapd
rm -rf db/*/*.db3
./bak2db /export/ds6/slapd-example/bak/2002_08_26_17_01_20
```

The pathname used in the `bak2db` command must match that produced by the `db2bak` command you already executed. If necessary, perform a directory listing of the `bak` directory to find the correct name.

Step 3: Execute the `start-slapd` and `ldapsearch` commands shown in Listing 4.9 to restart the server and perform a quick one-level search to verify that the data has been restored. The commands and output shown are from a Windows system. The output indicates that the restore was successful.

Listing 4.9 Checking That the Data Has Been Restored

```
start-slapd
C:\Netscape\Servers\slapd-example>net start slapd-example
The Netscape Directory Server 6 (example) service is starting.
The Netscape Directory Server 6 (example) service was started successfully.

cd \Netscape\Servers\shared\bin
ldapsearch -v -b "dc=example,dc=com" -s one "(objectClass=*)"
version: 1
dn: ou=Groups, dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: Groups

dn: ou=People, dc=example,dc=com
objectClass: top
objectClass: organizationalunit
ou: People

dn: ou=Special Users,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Special Users
description: Special Administrative Accounts

dn: ou=Netscape Servers,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Netscape Servers
description: Standard branch for Netscape Server registration
```

Controlling Access to Directory Data

Netscape Directory Server allows directory administrators to control access to all data in the DIT down to the entry, attribute, and value levels. Access control instructions are stored in operational attributes named `aci`. The `aci` attributes may appear in any entry, and by default they affect all entries within the subtree where they are stored. For example, access control instructions that are stored in the entry `dc=example,dc=com` govern access to all entries at and below `dc=example,dc=com`, such as those within `ou=People,dc=example,dc=com` and within `ou=Groups,dc=example,dc=com`.

This section introduces Netscape Directory Server's access control mechanism by demonstrating how to add an access control instruction to allow one entry to impersonate another. The actual impersonation is done using the LDAP Proxied Authorization control (see Chapter 3, LDAPv3 Extensions, for more information on this control). The Netscape access control mechanism uses a set of operation-specific rights to control access. To be able to use the Proxied Authorization control, the entry must have `proxy` rights for the entry you wish to impersonate.

Typically, the `proxy` right is granted to an administrative entry or to an entry that represents a software application. Follow these steps to add a special user to which you will grant the `proxy` right:

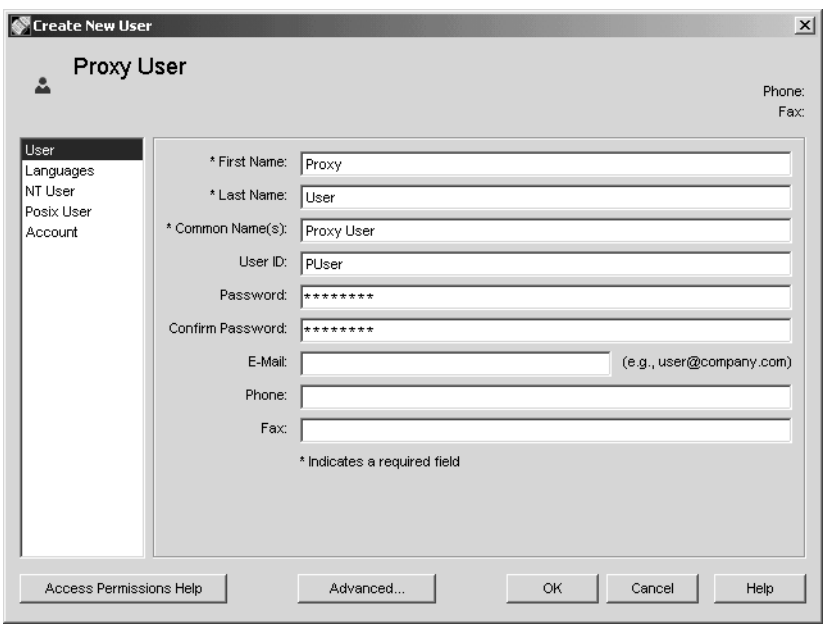
Step 1: Start Netscape Console if it is not already running, and log in, entering "cn=Directory Manager" as the DN and "secret389" as the password.

Step 2: Open the Directory Server Administration Console for the sample server by clicking the **Directory** tab, and click to expand the directory node labeled **example**. You should see four organizational unit entries: **Groups**, **People**, **Special Users**, and **Netscape Servers**.

Step 3: Select **Special Users** by clicking it. On the **Object** menu, choose the **New User** command. Create a user named "Proxy User" and give it the password "lrtw,YB!". Figure 4.7 shows how the **Create New User** screen should look before you click the **OK** button.

Step 4: Confirm that the new user is not yet able to act as a proxy for other entries. The tests performed rely on the fact that access control instructions to allow people to modify their own entry are included in the `Example.ldif` file. Listing 4.10

Figure 4.7 Creating a New Entry Named "Proxy User"



shows two `ldapmodify` commands that both authenticate as the Proxy User entry and attempt to change the `userPassword` attribute within Sam Carter's entry. The first command does not use Proxied Authorization; the second one does. The text you need to type is shown in bold. Press return twice to insert a blank line after the "`-`" character that appears on a line by itself. On Microsoft Windows, omit the leading "`./`" from the commands.

Listing 4.10 Failed Attempts to Modify an Entry

```
./ldapmodify -D "uid=puser,ou=Special Users,dc=example,dc=com" -w "lrtw,YB!"
dn: uid=scarter,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: mySecret42
-

modifying entry uid=scarter,ou=People,dc=example,dc=com
ldap_modify: Insufficient access
ldap_modify: additional info: Insufficient 'write' privilege to the
 ↳   'userPassword' attribute of entry
 ↳   'uid=scarter,ou=people,dc=example,dc=com'.

./ldapmodify -D "uid=puser,ou=Special Users,dc=example,dc=com" -w "lrtw,YB!"
 ↳  -Y "dn:uid=scarter,ou=People,dc=example,dc=com"
dn: uid=scarter,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: mySecret42
-

modifying entry uid=scarter,ou=People,dc=example,dc=com
ldap_modify: Insufficient access
ldap_modify: additional info: Insufficient 'write' privilege to the
 ↳   'userPassword' attribute of entry
 ↳   'uid=scarter,ou=people,dc=example,dc=com'.
```

The first `ldapmodify` command failed because the Proxy User entry is treated just like any other entry, and therefore it does not have permission to modify Sam Carter's entries. The second `ldapmodify` command failed because Proxy User does not yet have permission to impersonate other users, which makes using the `-Y` (Proxied Authorization) option unhelpful. If the server allowed Proxy User to impersonate Sam Carter, Proxy User would be able to modify Sam's entry (just as Sam Carter himself could).

Next, follow these steps to begin the process of adding a new access control instruction (ACI) to the `ou=People,dc=example,dc=com` subtree:

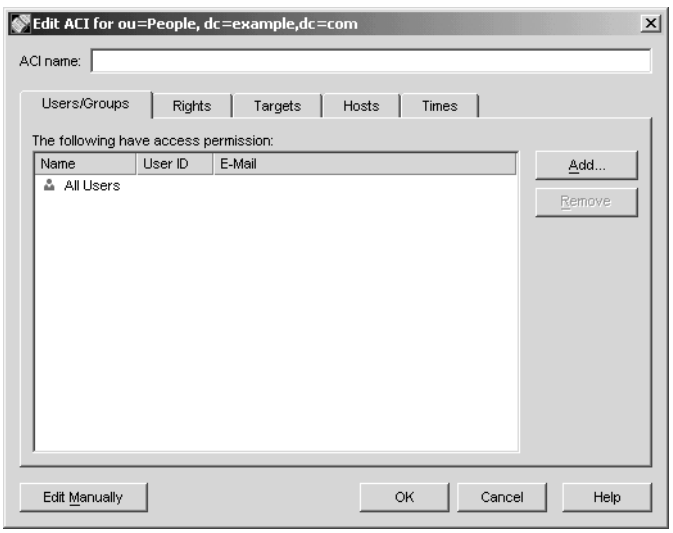
Step 1: Return to the Netscape Directory Server console **Directory** tab and select the node labeled **People**.

Step 2: From the **Object** menu, choose the **Set Access Permissions** command. A **Manage Access Control** window will open. It shows a list of five access control instructions, which are included in the `Example.ldif` file.

Step 3: Click the **New** button. Figure 4.8 shows the **Edit ACI** window that opens. This window has an ACI **Name** field, as well as five tabs:

1. **Users/Groups**. Allows you to add users and groups that are given the rights granted by this access control instruction. By default, the set of users and groups is set to **All Users**.
2. **Rights**. Allows you to specify what the users and groups are allowed to do. The available rights are these:
   - **`read`**. See attribute values, for example, by asking that an attribute be returned from an LDAP search operation.
   - **`compare`**. Compare attribute values, for example, by using an LDAP compare operation.
   - **`search`**. Determine if attribute values exist, for example, by using an attribute within an LDAP search filter.
   - **`selfwrite`**. Allow an entry to add its own DN to an attribute.
   - **`write`**. Modify attributes, for example, by using an LDAP modify operation.
   - **`delete`**. Remove entries by using an LDAP delete operation.
   - **`add`**. Add entries by using an LDAP add operation.

Figure 4.8 The Netscape Directory Server Console Edit ACI Window



- **proxy**. Impersonate another entry by using the LDAPv3 Proxied Authorization control.

By default, the new ACI grants all rights except `proxy`.

3. **Targets**. Allows you to limit the set of entries and attributes that this access control instruction governs. For example, you can specify that an ACI governs only the `userPassword` attribute of entries that match the filter `(objectClass=inetOrgPerson)`. By default, the ACI affects all entries at and below the entry that contains the `aci` attribute, and all attributes within those entries.
4. **Hosts**. Allows you to limit access based on the LDAP client's host name or IP address. By default, all hosts are treated the same.
5. **Times**. Allows you to limit access based on time of day and day of the week. For example, you could limit access to the hours 8 A.M. to 6 P.M. on weekdays. By default, no time- or day-based restrictions are enforced.

Next, follow these steps to grant the `proxy` right to Proxy User:

Step 1: Type the phrase "Proxy permission for Proxy User" in the ACI **Name** text field.

Step 2: Make sure the **Users/Groups** tab is active and that **All Users** is selected. Click the **Remove** button to delete "All Users" and then click the **Add** button to open the **Add Users and Groups** window. Search for the Proxy User entry and add it to the access permission list. Click the **OK** button to close the **Add Users and Groups** window.

Step 3: Click the **Rights** tab and make sure **Proxy** is the only right checked (you must uncheck all the other rights and then check **Proxy**). Click the **OK** button to save your new access control instruction. Figure 4.9 shows the updated **Manage Access Control** window that includes your new proxy permission ACI.

Step 4: Using Netscape Console to manage access control simplifies the process considerably and allows you to avoid the messy syntax of the `aci` attributes. Use the `ldapsearch` command shown in Listing 4.11 to list the `aci` values present in the `ou=People,dc=example,dc=com` entry.

Listing 4.11 Examining `aci` Values from the Command Line

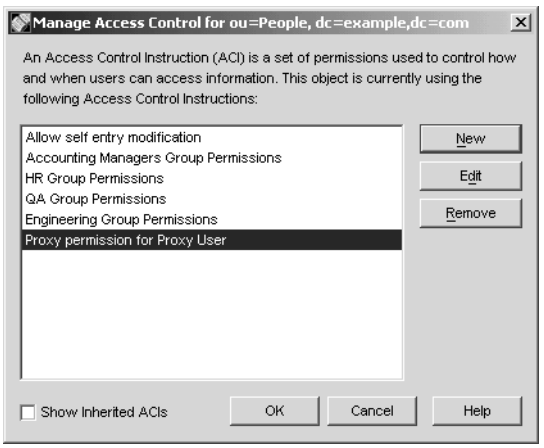
```
./ldapsearch -b "ou=People,dc=example,dc=com" -s base "(objectClass=*)" aci
version: 1
dn: ou=People, dc=example,dc=com
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr
 ↳  ="userpassword
  || telephonenumber || facsimiletelephonenumber")(version 3.0;acl "Allow
 ↳  self entry modification";allow (write)(userdn = "ldap:///self");)
```

```
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !="cn || sn ||
  uid")(targetfilter ="(ou=Accounting)")(version 3.0;acl "Accounting Managers
  Group Permissions";allow (write) (groupdn = "ldap:///cn=Accounting Managers
 ,ou=groups,dc=example,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !="cn || sn ||
  uid")(targetfilter ="(ou=Human Resources)")(version 3.0;acl "HR Group Permi
  ssions";allow (write)(groupdn = "ldap:///cn=HR Managers,ou=groups,dc=example
 ,dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !="cn ||sn ||
 uid")(targetfilter ="(ou=Product Testing)")(version 3.0;acl "QA Group Permis
 sions";allow (write)(groupdn = "ldap:///cn=QA Managers,ou=groups,dc=example,
 dc=com");)
aci: (target ="ldap:///ou=People,dc=example,dc=com")(targetattr !="cn || sn ||
  uid")(targetfilter ="(ou=Product Development)")(version 3.0;acl "Engineerin
  g Group Permissions";allow (write)(groupdn = "ldap:///cn=PD Managers,ou=grou
  ps,dc=example,dc=com");)
aci: (targetattr = "*") (version 3.0;acl "Proxy permission for Proxy User";all
  ow (proxy)(userdn = "ldap:///uid=PUser,ou=Special Users,dc=example,dc=com");
  )
```

The ACI that was just added is the last one. It targets all attributes (`*`) and grants the proxy right to the entry with the DN `uid=PUser,ou=Special Users, dc=example,dc=com`. That sounds correct, although you're probably glad you did not have to type it yourself.

Step 5: Use an `ldapmodify` command to impersonate Sam Carter and modify his entry. Listing 4.12 shows the command and the result. The entry modification was a success; the same command that failed earlier worked this time.

Figure 4.9 The Manage Access Control Window for `ou=People,dc=example,dc=com`



Listing 4.12 Using Proxied Authorization to Modify an Entry

```
./ldapmodify -v -D "uid=puser,ou=Special Users,dc=example,dc=com" -w "lrtw,YB!"
↳ -Y "dn:uid=scarter,ou=People,dc=example,dc=com"
ldapmodify: started Wed Aug 27 12:24:35 2002

ldap_init( localhost, 389 )
dn: uid=scarter,ou=People,dc=example,dc=com
changetype: modify
replace: userPassword
userPassword: mySecret42
-

replace userPassword:
        mySecret42
modifying entry uid=scarter,ou=People,dc=example,dc=com
modify complete
<Ctrl-C>
```

You can learn more about Netscape Directory Server's access control features by reading Netscape's documentation.

Changing the Server Configuration Using LDAP

Netscape Directory Server exposes all of its configuration as a series of directory server–specific entries that reside within a subtree named `cn=config`. The directory server's configuration entries and attributes are documented in Netscape's *Directory Server Configuration, Command, and File Reference* manual. You can examine all of the stored configuration entries and attributes by viewing the `config/dse.ldif` file within the server instance directory. You can change the configuration using Netscape Console, by stopping the server and editing the `config/dse.ldif` file, or by using LDAP modify operations that target the entries within the `cn=config` subtree.

This section demonstrates how to change a configuration setting using LDAP. Specifically, you will change a setting so that user data can no longer be updated; that is, LDAP add and modify operations will be rejected. Changing configuration settings using LDAP is useful when you're writing automated scripts that help manage a directory service deployment. The setting that you will learn how to change using the ldapmodify command-line utility can also be changed using the Netscape Directory Server console. To find the setting that controls whether the `userRoot` database will accept or reject LDAP updates, follow these steps:

Step 1: Open the Directory Server console and click on the **Configuration** tab.

Step 2: Click on the plus sign next to the **Data** node to reveal its contents. Several nodes will be visible, including **Database Link Settings**, **Database Settings**, and **dc=example,dc=com**.

Step 3: Click on the plus sign next to the **dc=example,dc=com** node to show its contents. One node named **userRoot** should be visible.

Step 4: Click on the **userRoot** node and then on the **Database Settings** tab on the right-hand side of the console window. The setting you're looking for is now visible as a check box labeled **Database is read-only**. By default, this is not checked; if it is checked, LDAP operations that change data are rejected by the server.

However, this section shows you how to change the setting without using the console. Follow these steps:

Step 1: Start a Unix shell or a Microsoft Windows command prompt window. On Solaris, type this command:

```
cd /export/ds6/shared/bin
```

On Microsoft Windows, type this command:

```
cd \Netscape\Servers\shared\bin
```

If you're using the Windows command prompt, omit the leading "`./`" sequences from the commands that follow.

The attribute that controls whether a database instance is writable is named `nsslapd-readonly`, and for the default user database instance it is located in the configuration entry named `cn=userRoot,cn=ldbm database,cn=plugins,cn=config` (*LDBM* stands for LDAP Database Manager and is the general name for Netscape's built-in LDAP data store). Entry updates are allowed by default, so the value of `nsslapd-readonly` is `off` for all database instances; you will change the value within the `userRoot` configuration entry to `on` in order to disable updates for that database instance.

Step 2: Execute the two `ldapmodify` commands shown in Listing 4.13 to change the `nsslapd-readonly` setting to `on` and to test whether updates were indeed disabled. For demonstration purposes, the first `ldapmodify` command modifies the `nsslapd-readonly` configuration attribute while authenticated as the Directory Manager entry. That entry has full privileges within Netscape Directory Server; however, the `cn=config` subtree does support fine-grained access control using the same mechanism as the rest of the Netscape server.

Listing 4.13 Disabling Updates by Modifying a Configuration Entry

```
./ldapmodify -D "cn=Directory Manager" -w secret389
dn: cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-readonly
nsslapd-readonly: on
-
```

```
modifying entry cn=userRoot,cn=ldbm database,cn=plugins,cn=config
<Ctrl-C>

./ldapmodify -D "uid=kvaughan,ou=People,dc=example,dc=com" -w bribery
dn: uid=dmiller,ou=People,dc=example,dc=com
changetype: modify
replace: cn
cn: Dave Miller
cn: David Miller
-

modifying entry uid=dmiller,ou=People,dc=example,dc=com
ldap_modify: DSA is unwilling to perform
ldap_modify: additional info: database is read-only
```

The second `ldapmodify` command authenticates as Kirsten Vaughan (`kvaughan`) and attempts to modify the `cn` attribute values within David Miller's entry (`dmiller`). The `kvaughan` entry is part of a Directory Administrators group that has full access to the entries under the `ou=People,dc=example,dc=com` subtree. Kirsten's password in the `Example.ldif` data is "bribery."

Because the David Miller entry is in the `userRoot` database that has been configured to reject updates, the command fails with the error "DSA is unwilling to perform."

Step 3: Execute the command shown in Listing 4.14 to restore the original configuration setting.

LDAP as a Server Administration Protocol

Exposing an extensive collection of server or application configuration information via LDAP is unusual, but this approach works well for Netscape Directory Server. LDAP is an open protocol that enables remote administration and allows a variety of configuration tools to be developed. Netscape Console communicates with the directory server via LDAP, as do many of Netscape's command-line utilities and scripts. The directory server can check the syntax and range of configuration values before accepting a change, and its powerful access control features can be used to regulate access to the configuration data. In addition, configuration changes take effect instantly; there is no need to restart the server or tell it to read a configuration file.

One potential disadvantage of using LDAP as a server administration protocol is that if intruders are able to get past the LDAP server's access control protection, they can reconfigure the server—but a similar risk exists with any method that supports remote administration.

Listing 4.14 Reenabling Updates by Modifying a Configuration Entry

```
./ldapmodify -D "cn=Directory Manager" -w secret389
dn: cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-readonly
nsslapd-readonly: off
-

modifying entry cn=userRoot,cn=ldbm database,cn=plugins,cn=config
<Ctrl-C>
```

Product Focus and Feature Set

Now that you have completed a brief hands-on tour of Netscape Directory Server, it is time to park the car for a while and skim the owner's manual. This section describes the origin, focus, and feature set of the Netscape server with an eye toward introducing you to some common characteristics of LDAP directory service products.

Origin

The first Netscape LDAP product, Netscape Directory Server 1.0, was delivered to the marketplace in September 1996. But Netscape did not start from scratch; it based its product on the open-source LDAP version 2 (LDAPv2) implementation from the University of Michigan. Version 1.0 of the Netscape product supported LDAPv2, server-to-server replication, a sophisticated access control scheme, a synchronization agent for Microsoft Windows NT domain directories, and an HTML template–based HTTP-to-LDAP gateway. Netscape quickly added LDAP support to the rest of its enterprise software, including products such as Netscape Communicator and Netscape SuiteSpot (a collection of integrated servers that included Netscape Enterprise Server, Netscape Messaging Server, and other servers). Netscape also developed software development kits (SDKs) and tools for LDAP developers.

Netscape shipped two major releases over the next three years, adding features such as LDAPv3 support, Netscape Console, and more robust replication features. In March 1999, Netscape and Sun Microsystems entered into the Sun-Netscape Alliance, in which the two companies jointly developed a variety of enterprise server products and delivered them under the iPlanet brand name. Later, Sun acquired Innosoft International, an open standards directory and messaging software company. The best ideas from Sun's own LDAP server (SunDS) and Innosoft's LDAP server (IDDS) were fed into the development of the product that was originally planned as Netscape Directory Server 5.0. The combined product shipped in May 2001 as iPlanet Directory Server 5.0, and it included

noteworthy features such as multimaster replication, server-to-server chaining, entry distribution, and role-based access control.

As the Sun-Netscape Alliance was coming to its scheduled end, Netscape and Sun decided to go their separate ways. Sun continues to develop the iPlanet line of server products (now sold under the "Sun ONE" moniker), and Netscape again develops its own line of server products. In December 2001, Netscape shipped Netscape Directory Server 6.0, which includes support for DSML, as well as integration with some America Online services, such as AOL Instant Messenger (America Online is Netscape's parent company).

Product Focus

Netscape Directory Server is designed primarily to address the needs of large enterprises, e-commerce companies, and extranets. Netscape has historically been a performance and scalability leader. The Netscape product supports millions of LDAP entries per server and can process thousands of simple search operations per second. High performance for add and modify operations is not Netscape's focus; Netscape chooses to focus on search performance somewhat at the expense of update performance.

The Netscape server is designed to meet the needs of applications that work with a logically centralized directory service. Netscape has a broad line of LDAP-enabled products, including

- **Netscape Communicator** and **Netscape 7.0**. Web browser and e-mail client suites.
- **Netscape Certificate Management System**. A flexible, standards-based public key infrastructure (PKI) server suite that supports certificate issue, renewal, suspension, revocation, and online status checks.
- **Netscape Enterprise Server**. A high-performance, secure Web application server that is used by many high-traffic Web sites. Enterprise Server can use LDAP for authentication and access control.
- **AIM Enterprise Gateway**. An AOL Instant Messenger (AIM) gateway that allows AIM use to be managed by an enterprise. The AIM Enterprise Gateway acts as a proxy between users inside the corporate firewall and those on AOL's public AIM network, enabling enterprises to manage and control how employees use AIM services. LDAP-enabled identity management features map AIM screen names to corporate employee IDs and group employees by job function or department.
- **Netscape Delegated Administrator**. A product that builds on Netscape Directory Server to provide a Web-based interface for delegated directory and services administration

as well as end-user self-administration. This product supports many levels of delegation of authority and many types of directory and service administrators (for example, e-mail administrators).

In addition, leading application and middleware vendors such as Netegrity, Hewlett-Packard, and IBM support Netscape Directory Server in their own LDAP-enabled products. Netscape Directory Server is a multiplatform product that runs on several leading server hardware and software platforms. Netscape also supports hardware-based accelerators to improve Secure Sockets Layer (SSL) and Transport Layer Security (TLS) performance.

The Netscape server is a flexible product that is easy to deploy and manage. It is therefore used in deployments that range from one location with a single server to large, multinational companies with thousands of LDAP servers. Netscape does not focus on the needs of network operating systems (where authentication, file services, and printing services still dominate the requirements). However, the Netscape product is popular with Unix operating system vendors. For example, Hewlett-Packard bundles the Netscape product with its HP/UX operating system to provide authentication, authorization, and centralized storage for the OS and its applications.

Feature Set

The Netscape product provides a broad set of features. Although some of the features are specific to this product, many of them are supported by other leading LDAPv3 servers. The Netscape feature set includes

- Support for LDAPv3 standards, including RFCs 2251, 2252, 2253, 2254, 2255, 2829, and 2830. This feature provides interoperability with LDAP clients and servers from other vendors, as well as strong, standards-based security.
- Support for many LDAPv3 controls, including those that provide Virtual List View, server-side sorting, persistent searching, proxied authorization, `ManageDSAIT`, and password expiration. Some of these are proposed standards, and some are not; see Chapter 3, LDAPv3 Extensions, for more information on LDAPv3 controls.
- Support for many LDAPv3 extended operations, including StartTLS and online bulk import. See Chapter 3, LDAPv3 Extensions, for more information on LDAPv3 extended operations.
- Entry distribution to allow entries within one subtree of the DIT to be stored on more than one server. Distribution can be based on the location of an entry within a subtree, or a custom distribution algorithm can be specified; for example, a one-way hash of each entry's DN might be used.

- Extensible schema and configurable attribute indexes to support custom applications. New schema information can be added over LDAP. Netscape Directory Server 6 supports one subschema subentry per server; there is no support for using different schemas for different portions of the DIT.
- Server-to-server replication over LDAPv3. This feature provides flexible replication schedules, multiple writable copies (multiple masters), cascaded replication (chains of replicas), and replication monitoring.
- Flexible, scalable directory access control in which the rules are stored in the DIT so that they can be managed by means of LDAP and replicated with the regular directory data. Access control can be based on a DN, simple or dynamic (filter-based) groups, roles, time of day, day of the week, IP address, and other criteria. Fine-grained access control can be applied to entries, attributes, and attribute values.
- Support for several local directory databases on one server, or several remote databases located on network-connected servers (through server-to-server LDAP chaining). The databases support high-speed import from a file (LDIF or DSML format) and online bulk import over LDAP through a proprietary LDAPv3 extended operation.
- Data integrity features, including a transactional data store, referential integrity, attribute uniqueness enforcement, and the ability to restrict attribute values to ASCII (7-bit) characters.
- A server plug-in API that allows developers outside of Netscape to extend the functionality of the directory server in an arbitrary manner. For example, plug-in writers can implement preoperation filters, postoperation triggers, and new controls and extended operations. See the next section, Extending the Netscape Server: A Simple Plug-in Example, for more information on plug-ins.
- Integration with AOL's Instant Messenger (AIM) service to allow retrieval over LDAP of status information, such as whether someone is connected to AIM. This feature is unique to the Netscape server.
- A class of service (CoS) feature that provides collective or shared attribute values. Using CoS, you can manage an attribute value in one place that appears in thousands of directory entries. Although CoS is unique to the Netscape and Sun servers, some other directory servers also support shared attribute values.
- Dynamic groups and roles to provide convenient ways to manage collections of entries. Dynamic groups and roles may be used for access control, to control CoS attributes, and by LDAP applications.

- Support for SSL and TLS, including X.509v3 certificate–based authentication and the ability to algorithmically map information in a certificate to a specific directory entry. Netscape also supports hardware-based accelerators to improve SSL and TLS performance.
- Server monitoring using Simple Network Management Protocol (SNMP) and LDAP.
- Tools for C, C++, Java, JavaScript, and XML developers and for anyone using another language that can call C or Java code (such as Visual Basic). The Netscape SDKs support all popular OS and hardware platforms.

In summary, Netscape Directory Server 6 is a high-performance, highly scalable LDAP server that provides many features.

Extending the Netscape Server: A Simple Plug-in Example

Sometimes the cars available from the major automobile manufacturers simply do not meet all your needs. Or perhaps you just enjoy getting your hands dirty and cannot resist adding a turbocharger or reprogramming your engine computer. Some people need or want to make analogous enhancements to their server software. As mentioned in the preceding Feature Set section, Netscape Directory Server provides a plug-in API to allow developers outside of Netscape to extend the functionality of the server.

A plug-in is a Unix shared object or a Microsoft Windows DLL that presents a C interface to the directory server. Most plug-ins are written in C or C++. Plug-ins can be used to implement preoperation filters, postoperation triggers, new extended operations, new controls, and other extensions of the LDAP server functionality. Netscape uses plug-ins to provide some of the functionality of its server; therefore, many plug-ins are installed by default. Figure 4.10 shows the plug-ins configuration window within Netscape Console, with Netscape's Referential Integrity plug-in selected.

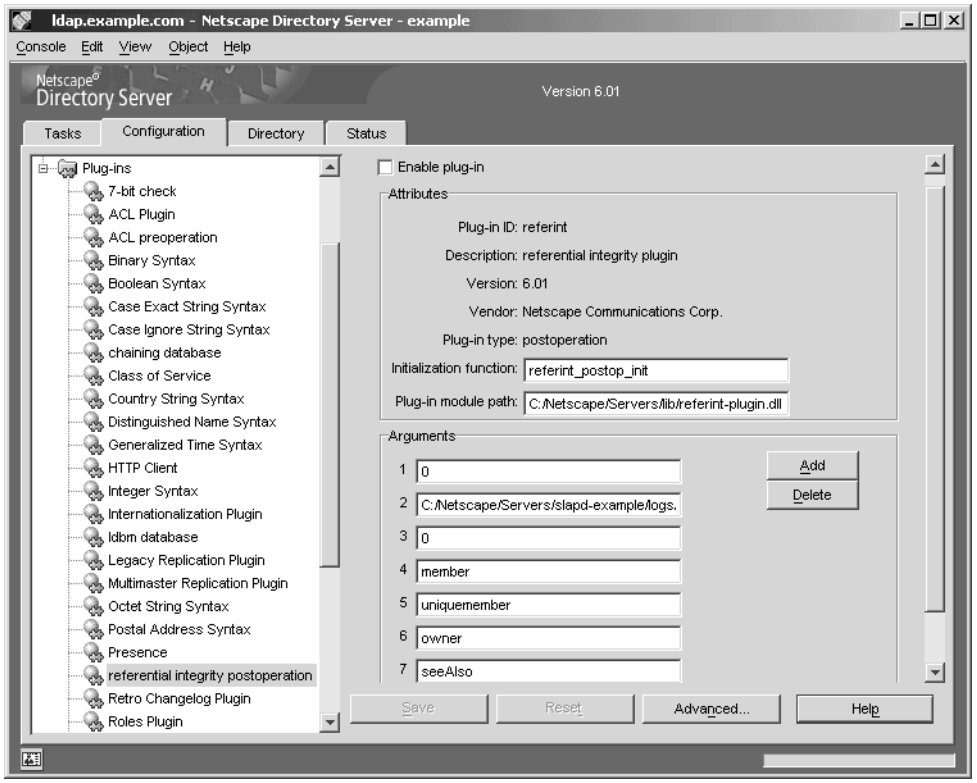
Each plug-in is managed through a plug-in configuration entry that resides in a special area within the DIT. For example, the configuration for Netscape's Referential Integrity plug-in is stored in an entry named `cn=referential integrity postoperation, cn=plugins,cn=config`. Listing 4.15 shows a sample configuration entry for the Referential Integrity plug-in (in LDIF format).

Listing 4.15 A Plug-in Configuration Entry

```
dn: cn=referential integrity postoperation,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
```

```
objectClass: extensibleObject
cn: referential integrity postoperation
nsslapd-pluginPath: C:/Netscape/Servers/lib/referint-plugin.dll
nsslapd-pluginInitfunc: referint_postop_init
nsslapd-pluginType: postoperation
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: 0
nsslapd-pluginarg1: C:/Netscape/Servers/slapd-example/logs/referint
nsslapd-pluginarg2: 0
nsslapd-pluginarg3: member
nsslapd-pluginarg4: uniquemember
nsslapd-pluginarg5: owner
nsslapd-pluginarg6: seeAlso
nsslapd-pluginarg7: nsroledn
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: referint
nsslapd-pluginVersion: 6.01
nsslapd-pluginVendor: Netscape Communications Corp.
nsslapd-pluginDescription: referential integrity plugin
```

Figure 4.10 The Netscape Console Plug-ins Configuration Window



The remainder of this section demonstrates how to develop and use a custom plug-in. Complete information on writing plug-ins can be found in the *Netscape Directory Server Plug-in Programmers' Guide*.

Problem Statement

Netscape Directory Server does not provide a way to limit the values of an attribute to a predefined set. For example, the `inetOrgPerson` object class defined in RFC 2798 allows person entries to include an `employeeType` attribute. Typically only a few values are supposed to be used for this attribute; however, the set of valid values is specific to each directory deployment. Although LDAP clients can be written so that they limit the set of values that users and directory data administrators may enter, it would be nice if the directory server itself were able to reject invalid values. The custom plug-in presented in this part of the chapter is named the *Value Constraint plug-in*. It extends the Netscape server so that LDAP add and modify operations that include invalid `employeeType` values are rejected, with an appropriate error message sent back to the LDAP client.

The Design of the Value Constraint Plug-In

To restrict `employeeType` values to a predefined set, the plug-in must be invoked before an LDAP add or modify operation is processed by the server. Therefore, the Value Constraint plug-in is implemented as a preoperation plug-in. Preoperation plug-ins can be used to alter operation parameters, to reject an operation entirely, or just to make note of something and allow the Netscape server to process the operation as usual.

The valid set of `employeeType` values is assumed to be

- Contractor
- Employee
- Intern
- Temp

The plug-in will check the `employeeType` values sent in LDAP add and modify operations to ensure that they match one of these values. If not, a "constraint violation" error will be returned to the LDAP client, and the server will not process the operation. The "constraint violation" error is a standard LDAP error that indicates that an attribute value is too large, is in the wrong format, or otherwise does not meet constraints imposed by the LDAP information model or by a local policy.

Note

Netscape Directory Server pre- and postoperation plug-ins are executed only when the server is processing regular LDAP operations. They are not executed for special tasks such as `ldif2db` (which bulk-loads an LDIF file into a database instance). Therefore, the Value Constraint plug-in cannot enforce any constraints during data import; you need to use a different method to do so.

The Source Code for the Value Constraint Plug-In

The Value Constraint plug-in source code consists of three files: `valueconstraint.c`, `dllmain.c`, and `valueconstraint.def`. The latter two files are required on Microsoft Windows only; the `dllmain.c` code is not shown in this book because it is simply copied from the sample plug-in code that Netscape ships with its server. The `dllmain.c` file you need to copy can be found in this location (assuming you installed the server in the default location as recommended earlier in this chapter):

```
C:\Netscape\Servers\plugins\slapd\slapi\examples\dllmain.c
```

Listing 4.16 shows the contents of the `valueconstraint.def` file, which is a Microsoft Windows DLL definition file.

Listing 4.16 The `valueconstraint.def` File

```
; Microsoft Windows DLL definition file for the valueconstraint
;    Netscape Directory Server 6 plugin.
;
; From the book "Understanding and Deploying LDAP Directory Services"
;   by Timothy A. Howes, Mark C. Smith, and Gordon S. Good.
;
DESCRIPTION     'valueconstraint plugin for Netscape Directory Server'
EXPORTS
    valueconstraint_init       @1
```

The `valueconstraint_init()` function is the only function that is exported from the Value Constraint plug-in DLL. It is the main entry point to the Value Constraint plug-in, and the directory server will be configured to call this function during server startup.

The plug-in's interesting code is written in C and is housed in a file named `valueconstraint.c`. Listing 4.17 shows the first portion of that file.

Listing 4.17 The Beginning of the `valueconstraint.c` File

```
 1. /*
 2.  * A valueconstraint plug-in for Netscape Directory Server
 3.  *
 4.  * From the book "Understanding and Deploying LDAP Directory Services"
 5.  *  by Timothy A. Howes, Mark C. Smith, and Gordon S. Good
 6.  */
 7. #include <string.h>
 8. #include <stdio.h>
 9. #include "slapi-plugin.h"
10.
11. /* Name and result code macros (to make the code easier to read) */
12. #define PLUGIN_NAME             "valueconstraint"
13. #define PLUGIN_RC_ERROR         (-1)
14. #define PLUGIN_RC_CONTINUE      0
15. #define PLUGIN_RC_RESULT_SENT   1
16.
17. /* Macros to take care of platform specifics */
18. #ifdef _WIN32
19. #define PLUGIN_STRCASECMP           stricmp
20. #define PLUGIN_EXPORTED_FUNCTION    __declspec(dllexport)
21. #else
22. #define PLUGIN_STRCASECMP           strcasecmp
23. #define PLUGIN_EXPORTED_FUNCTION
24. #endif
25.
26. /* Static variables to hold the plug-in name and description */
27. static Slapi_PluginDesc pdesc = { PLUGIN_NAME,
28.         "Howes/Smith/Good", "2.0", PLUGIN_NAME " plugin" };
29.
30. /* Static variables to hold attribute constraint configuration */
31. static const char *attr_to_check = "employeeType";
32. static const char *valid_values[] = {
33.     "Contractor", "Employee", "Intern", "Temp", NULL
34. };
35.
36. /* Function prototypes */
37. static int valueconstraint_preadd( Slapi_PBlock *pb );
38. static int valueconstraint_premodify( Slapi_PBlock *pb );
39. static int valueconstraint_is_one_of( Slapi_PBlock *pb,
40.         const char *attrname, const char *val,
41.         const char *allowed[] );
42.
```

Lines 1 through 25 contain comments, `include` statements, and some simple macro definitions. The main header file for the Netscape Directory Server plug-in interface is called `slapi-plugin.h`; it is included on line 9. A plug-in description structure is defined on lines 26 through 28; later that structure is registered with the directory server so that descriptive information about the plug-in is available through Netscape Console and elsewhere.

In this example the attribute and the set of valid values allowed are hard-coded into the `valueconstraint.c` file (a good alternative would be to store this information in a directory entry such as the plug-in's own configuration entry). Lines 30 through 34 define an `attr_to_check` variable (`employeeType`) and a NULL-terminated `valid_values` array. Prototypes for the static functions that are internal to the plug-in implementation appear on lines 36 through 41.

Listing 4.18 shows the `valueconstraint_init()` function that is called by the directory server during server startup. The purpose of this function is to perform any required initialization and to register plug-in callback functions with the directory server.

Listing 4.18 The `valueconstraint_init()` Function

```
43. /* Function valueconstraint_init(): initialization (called during
44.  *  server startup).
45.  */
46. PLUGIN_EXPORTED_FUNCTION int
47. valueconstraint_init( Slapi_PBlock *pb )
48. {
49.     int     rc = PLUGIN_RC_CONTINUE;
50.
51.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME, "=> init\n" );
52.
53.     /*
54.      * Register the plug-in version, plug-in description,
55.      * and two preoperation functions.
56.      */
57.     if ( slapi_pblock_set( pb, SLAPI_PLUGIN_VERSION,
58.             SLAPI_PLUGIN_VERSION_01 ) != 0 ||
59.             slapi_pblock_set( pb, SLAPI_PLUGIN_DESCRIPTION,
60.             &pdesc ) != 0 ||
61.             slapi_pblock_set( pb, SLAPI_PLUGIN_PRE_ADD_FN,
62.             (void *)valueconstraint_preadd ) != 0 ||
63.             slapi_pblock_set( pb, SLAPI_PLUGIN_PRE_MODIFY_FN,
64.             (void *)valueconstraint_premodify ) != 0 ) {
65.         slapi_log_error( SLAPI_LOG_FATAL, PLUGIN_NAME,
66.                 "init: a slapi_pblock_set() call failed\n" );
67.         rc = PLUGIN_RC_ERROR;
68.     }
69.
70.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME,
71.             "<= init (%d)\n", rc );
72.     return rc;
73. }
74.
```

The Netscape Directory Server plug-in API is sometimes referred to as the SLAPI API (which loosely stands for "standalone LDAP server API"), and most of the functions provided by Netscape have names that begin with "slapi". On line 51 in Listing 4.18, a

call is made to `slapi_log_error()`, which is a utility function that writes a message to the directory server's error log if the debug level provided in the first parameter (`SLAPI_LOG_PLUGIN`) is enabled in the server's configuration. By default, `SLAPI_LOG_PLUGIN` messages are not logged, but `SLAPI_LOG_FATAL` messages (used on line 65) are.

Information is passed back and forth between plug-ins and the core of the Netscape server through a parameter block data structure named `Slapi_PBlock` that can hold a variety of items. In fact, the only parameter passed to the `valueconstraint_init()` function is a pointer to a `Slapi_PBlock` data structure. The following two functions are used to get and set items within a SLAPI parameter block:

1. **`slapi_pblock_get()`**. Retrieves the value of an item from a `Slapi_PBlock` data structure. The item is identified by a macro whose name begins with `SLAPI_PLUGIN`, and the value of the item requested is returned via the last parameter, which is a pointer to an area to hold the value.

2. **`slapi_pblock_set()`**. Sets the value of an item in a `Slapi_PBlock` data structure. This function takes the same parameters as `slapi_pblock_get()`, except the last parameter is the value to set (not a pointer to the value).

The `slapi_pblock_get()` and `slapi_pblock_set()` functions are used often in the writing of SLAPI plug-ins. The code on lines 53 through 68 in Listing 4.18 includes four calls to the `slapi_pblock_set()` function to register four things: the API version used by the plug-in (`SLAPI_PLUGIN_VERSION`), the plug-in description structure (`SLAPI_PLUGIN_DESCRIPTION`), and two callback functions that will be called before each add or modify operation is executed in the server (`SLAPI_PLUGIN_PRE_ADD_FN` and `SLAPI_PLUGIN_PRE_MODIFY_FN`). A preoperation plug-in such as the Value Constraint plug-in can register one preoperation callback function for each kind of LDAP operation (bind, add, modify, delete, and so on). In this plug-in, there are only two such callback functions:

1. **`valueconstraint_preadd()`**. Called before the server executes each LDAP add operation but after the entire request has been received from the LDAP client and the request parameters have been parsed and placed in the `Slapi_PBlock` data structure. This function enforces the desired constraints on the `employeeType` attribute.

2. **`valueconstraint_premodify()`**. Called before the server executes each LDAP modify operation but after the entire request has been received from the LDAP client and the request parameters have been parsed and placed in the `Slapi_PBlock` data structure. This function also enforces constraints on the `employeeType` attribute.

Preoperation callback functions return an integer value that is examined by the Netscape server to determine if the operation should continue to execute or if processing should halt. In the latter case, the plug-in is responsible for sending an LDAP result message to the client. The two return values typically used are

- **0**. The server should continue executing the operation.
- **1**. The plug-in sends a result to the LDAP client, and the server should stop execution.

The macros defined on lines 14 (`PLUGIN_RC_CONTINUE`) and 15 (`PLUGIN_RC_RESULT_SENT`) of Listing 4.17 capture these special return values.

Listing 4.19 shows the code for the `valueconstraint_preadd()` function. It, too, is passed only one parameter, a `Slapi_PBlock` pointer. On lines 82 and 83, some additional Netscape-defined data types are used: `Slapi_Entry` (to represent a directory entry) and `Slapi_Attr` (to represent an attribute within an entry). Most of the data types defined by the Netscape SLAPI API, including these two and the `Slapi_PBlock` data structure introduced earlier, are *opaque* structures. This means that you can declare pointers to them, but you can't define them, determine their size, or look at their fields. Accessor functions are provided to manipulate these opaque data structures. This technique gives Netscape freedom to change the underlying implementation of its server without changing the SLAPI plug-in API.

Listing 4.19 The `valueconstraint_preadd()` Function

```
75. /* Function valueconstraint_preadd(): called by the directory
76.  *  server before an add operation is processed.
77.  */
78. static int
79. valueconstraint_preadd( Slapi_PBlock *pb )
80. {
81.     int         rc = PLUGIN_RC_CONTINUE;
82.     Slapi_Entry *entry = NULL;
83.     Slapi_Attr  *attr = NULL;
84.
85.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME, "=> preadd\n" );
86.
87.     /* Retrieve the entry that was sent in the add operation */
88.     if ( slapi_pblock_get( pb, SLAPI_ADD_ENTRY, &entry ) != 0 ) {
89.         slapi_send_ldap_result( pb, LDAP_OPERATIONS_ERROR, NULL,
90.                 PLUGIN_NAME ": unable to retrieve entry", 0, NULL );
91.         rc = PLUGIN_RC_RESULT_SENT;
92.     } else if ( slapi_entry_attr_find( entry, attr_to_check,
93.                 &attr ) == 0 ) {
94.         /* Check the attribute values to make sure they are valid */
95.         int         hint;
```

```
 96.         Slapi_Value *value;
 97.         const char  *strvalue;
 98.
 99.         for ( hint = slapi_attr_first_value( attr, &value );
100.                 hint != -1;
101.                 hint = slapi_attr_next_value( attr, hint, &value )) {
102.             strvalue = slapi_value_get_string( value );
103.             if ( !valueconstraint_is_one_of( pb, strvalue,
104.                     attr_to_check, valid_values )) {
105.                 rc = PLUGIN_RC_RESULT_SENT;
106.                 break;
107.             }
108.         }
109.     }
110.
111.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME,
112.             "<= preadd (%d)\n", rc );
113.     return rc;
114. }
115.
```

The `valueconstraint_preadd()` function first retrieves the entry that is being added (lines 87–91), and then it calls a `slapi_entry_attr_find()` `Slapi_Entry` accessor function to locate the attribute whose name is in the `attr_to_check` variable (which is `employeeType`). If that attribute is found, a `for` loop (lines 99–108) is used to examine each of the values and check that they are valid. Within the loop, the `slapi_attr_first_value()` and `slapi_attr_next_value()` functions are used to step through the values contained in the `Slapi_Attr` data structure, and the `slapi_value_get_string()` function is used to retrieve the string value. Finally, a utility function named `valueconstraint_is_one_of()` is used to determine if `strvalue` is in the set of valid values. If not, the function return code is set to `PLUGIN_RC_RESULT_SENT` (line 105). The `valueconstraint_is_one_of()` function sends an error result to the LDAP client if the value is not valid; the code for `valueconstraint_is_one_of()` is shown in Listing 4.21.

Listing 4.20 shows the code for the `valueconstraint_premodify()` function, which is quite similar to `valueconstraint_preadd()`. The code on lines 128 through 133 retrieves from the parameter block the list of modifications submitted by the LDAP client. The modifications are returned as a NULL-terminated array of pointers to `LDAPMod` structures. The `LDAPMod` structure is one of the few structures used in the Netscape plug-in API that is not opaque; it is actually part of the LDAP C API that is commonly used to write LDAP client applications.

Listing 4.20 The `valueconstraint_premodify()` Function

```
116. /* Function valueconstraint_premodify(): called by the directory
117.  *  server before a modify operation is processed.
118.  */
119. static int
120. valueconstraint_premodify( Slapi_PBlock *pb )
121. {
122.     int         rc = PLUGIN_RC_CONTINUE;
123.     LDAPMod     *mp, **mods = NULL;
124.     int         i, j;
125.
126.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME, "=> premodify\n" );
127.
128.     /* Retrieve the modifications sent in the modify operation */
129.     if ( slapi_pblock_get( pb, SLAPI_MODIFY_MODS, &mods ) != 0 ) {
130.         slapi_send_ldap_result( pb, LDAP_OPERATIONS_ERROR, NULL,
131.                 PLUGIN_NAME ": unable to retrieve modifications",
132.                 0, NULL );
133.         rc = PLUGIN_RC_RESULT_SENT;
134.     } else {
135.         /*
136.          * Check the modify suboperations to make sure the values
137.          *  they contain are valid. Delete suboperations are
138.          *  ignored (any value may be deleted).
139.          */
140.         for ( i = 0; rc == PLUGIN_RC_CONTINUE && mods[i] != NULL;
141.                     ++i ) {
142.             mp = mods[i];
143.             if ( 0 == slapi_attr_type_cmp( mp->mod_type,
144.                     attr_to_check, SLAPI_TYPE_CMP_BASE )
145.                     && LDAP_MOD_DELETE !=
146.                     ( mp->mod_op & ~LDAP_MOD_BVALUES )) {
147.                 for ( j = 0; mp->mod_bvalues[j] != NULL; ++j ) {
148.                     if ( !valueconstraint_is_one_of( pb,
149.                             mp->mod_bvalues[j]->bv_val,
150.                             attr_to_check, valid_values )) {
151.                         rc = PLUGIN_RC_RESULT_SENT;
152.                         break;
153.                     }
154.                 }
155.             }
156.         }
157.     }
158.
159.     slapi_log_error( SLAPI_LOG_PLUGIN, PLUGIN_NAME,
160.             "<= premodify (%d)\n", rc );
161.     return rc;
162. }
163.
```

The code on lines 135 through 156 loops through the modifications and checks each one to see if it is an "add value" or a "replace value" operation on the `employeeType` attribute (recall that `attr_to_check` was initialized to the constant string `"employeeType"`). The Netscape-provided `slapi_attr_type_cmp()` utility function is used to compare the attribute type name in the `LDAPMod` structures with `attr_to_check`. The `for` loop on lines 147 through 154 steps through the values, calling the `valueconstraint_is_one_of()` function for each one to check for invalid values.

Listing 4.21 shows the code for the `valueconstraint_is_one_of()` function, which returns an indication of whether a string value is in the array of allowed values. This function sends an error result to the LDAP client if an invalid value is detected.

Listing 4.21 The `valueconstraint_is_one_of()` Utility Function

```
164. /* Function valueconstraint_is_one_of(): return a nonzero value
165.  *  if "val" is in the NULL-terminated "allowed" array and 0 if not.
166.  *  Comparisons are case insensitive. If "val" is not present, a
167.  *  "constraint violation" error is sent to the LDAP client.
168.  */
169. static int
170. valueconstraint_is_one_of( Slapi_PBlock *pb, const char *val,
171.         const char *attrname, const char *allowed[] )
172. {
173.     int         i;
174.     char        *msg;
175.     const char  *fmt = "invalid value \"%s\" for %s";
176.
177.     if ( val == NULL ) {
178.         return 1;   /* ignore NULL values */
179.     }
180.
181.     for ( i = 0; allowed[i] != NULL; ++i ) {
182.         if ( PLUGIN_STRCASECMP( val, allowed[i] ) == 0 ) {
183.             return 1;   /* found it */
184.         }
185.     }
186.
187.     /* Not found: send back a "constraint violation" error */
188.     msg = slapi_ch_malloc( strlen(fmt) + strlen(val)
189.             + strlen(attrname) + 1 );
190.     sprintf( msg, fmt, val, attrname );
191.     slapi_send_ldap_result( pb, LDAP_CONSTRAINT_VIOLATION,
192.             NULL, msg, 0, NULL );
193.     slapi_ch_free_string( &msg );
194.     return 0;   /* not found */
195. }
```

The `valueconstraint_is_one_of()` function is the last function in `valueconstraint.c`. The code on lines 181 through 185 looks for `val` (the value) within the `allowed` array using a case-insensitive comparison function (`strcasecmp()` or `stricmp()`, abstracted away by the `PLUGIN_STRCASECMP()` macro). If the value is in the set of allowed values, a nonzero value is returned by line 183.

The code on lines 187 through 194 constructs a human-readable error message and uses the `slapi_send_ldap_result()` Netscape plug-in API utility function to send that message along with an LDAP "constraint violation" result code to the client. The `LDAP_CONSTRAINT_VIOLATION` macro is defined by the `ldap.h` header file that is part of the LDAP C API. The `ldap.h` file is included from `slapi-plugin.h`.

Compiling and Installing the Value Constraint Plug-In

On Microsoft Windows, execute the following three commands at the command prompt to compile the two `.c` files that make up the plug-in (using the Microsoft Visual C++ `cl` command) and create a DLL named `valueconstraint.dll` (using the Visual C++ `link` command):

```
cl -IC:\Netscape\Servers\plugins\slapd\slapi\include /c valueconstraint.c

cl -IC:\Netscape\Servers\plugins\slapd\slapi\include /c dllmain.c

link /dll /def:valueconstraint.def /out:valueconstraint.dll
↳  /DEFAULTLIB:kernel32.lib user32.lib gdi32.lib winspool.lib
↳  comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib
↳  uuid.lib odbc32.lib odbccp32.lib wsock32.lib
↳  C:\Netscape\Servers\plugins\slapd\slapi\lib\libslapd.lib
↳  C:\Netscape\Servers\plugins\slapd\slapi\lib\libnspr4.lib
↳  valueconstraint.obj dllmain.obj
```

Yes, the `link` command is very long. On Solaris, use these two commands to compile the `valueconstraint.c` file and create a shared library named `valueconstraint.so`:

```
cc -I/export/ds6/plugins/slapd/slapi/include -D_REENTRANT -KPIC -c
↳ valueconstraint.c

ld -G -o valueconstraint.so valueconstraint.o /export/ds6/lib/libslapd.so
↳ /export/ds6/lib/libnspr4.so
```

The commands for other platforms are similar; consult the Netscape documentation for details.

To install the plug-in, shut down the directory server using the `stop-slapd` command. Next, edit the `dse.ldif` file located in the server `config` directory within the file system

to add a configuration entry for the plug-in. Listing 4.22 shows the correct entry for a Solaris installation, assuming the Value Constraint shared library is located in a directory named `/usr/netscape-server-plugins`.

Listing 4.22 The Value Constraint Plug-in Configuration Entry

```
dn: cn=valueconstraint,cn=plugins,cn=config
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: valueconstraint
nsslapd-pluginPath: /usr/netscape-server-plugins/valueconstraint.so
nsslapd-pluginInitfunc: valueconstraint_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-plugin-depends-on-type: database
nsslapd-pluginId: valueconstraint
nsslapd-pluginVersion: 2.0
nsslapd-pluginVendor: Howes/Smith/Good
nsslapd-pluginDescription: valueconstraint plugin
```

For a Microsoft Windows installation, change `nsslapd-pluginPath` to point to your `valueconstraint.dll` file. After saving your changes to the `dse.ldif` file, restart the server. You can also add the plug-in configuration entry over LDAP while the directory server is running and then restart the server to load the plug-in.

Note

Adding new plug-ins always requires a restart of Netscape Directory Server. Through the use of Netscape Console or modification of the `nsslapd-pluginEnabled` attribute within a plug-in's configuration entry, plug-ins that were present when the server was last started can be disabled or enabled while the server is running. The value of `nsslapd-pluginEnabled` may be `off` or `on`.

Using Netscape Console to examine the list of active plug-ins, you can verify that your Value Constraint plug-in was recognized by the server and correctly loaded.

The Resulting Server Behavior

Verify that the plug-in is working correctly by trying some add and modify operations. Any LDAP client that supports those operations may be used.

Listing 4.23 shows an LDIF file that contains two entries. The first includes an `employeeType` value of `unknown`, which should be rejected by the Value Constraint plug-in as invalid. The second entry includes a valid value (`contractor`).

Note

The commands shown in this section assume that the entries from Netscape's `Example.ldif` file have been loaded into your server. If in doubt, execute a command like this to load the `Example.ldif` file:

```
./ldif2db -n userRoot -i - <ldif/Example.ldif
```

The `ldif2db` command must be executed from the `\Netscape\Servers\slapd-example` directory on Windows or from the `/export/ds6/slapd-example` directory on Solaris. The other commands in this section rely on the presence of some of the entries and `aci` attributes from the `Example.ldif` file.

Listing 4.23 Entries with Different `employeeType` Values

```
version: 1
# this add operation should fail with constraintViolation
dn: uid=cjones,ou=People,dc=example,dc=com
cn: Christina Jones
sn: Jones
givenName: Christina
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
ou: Accounting
ou: People
L: Sunnyvale
uid: cjones
mail: cjones@example.com
userPassword: secret
employeeType: unknown

# this add operation should succeed
dn: uid=cjones,ou=People,dc=example,dc=com
cn: Christina Jones
sn: Jones
givenName: Christina
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
ou: Accounting
ou: People
L: Sunnyvale
uid: cjones
mail: cjones@example.com
userPassword: secret
employeeType: contractor
```

Place the contents of Listing 4.23 in a file named `testadds.ldif` and use the `ldapmodify` command to test the Value Constraint plug-in's behavior when processing LDAP add operations. Listing 4.24 shows the command and the result. The `-a` option (add entries) and the `-c` option (continue even if an error occurs) are passed to `ldapmodify` so that the command will try to add the second entry even if the first add fails. The result indicates that the test was a success; the plug-in returned a "constraint violation" error.

Listing 4.24 Testing the Value Constraint Plug-in with Add Operations

```
./ldapmodify -ac -D "uid=kvaughan,ou=People,dc=example,dc=com" -w bribery
↳ < testadds.ldif

adding new entry uid=cjones,ou=People,dc=example,dc=com
ldap_add: Constraint violation
ldap_add: additional info: invalid value "unknown" for employeeType

adding new entry uid=cjones,ou=People,dc=example,dc=com
```

Next, test the behavior of the Value Constraint plug-in when it is confronted with LDAP modify operations. Listing 4.25 shows an LDIF file that contains a series of modify operations. Comments are included in the file to indicate whether the operation should succeed or not.

Listing 4.25 Modify Operations that Use Different `employeeType` Values

```
version: 1
# this modify operation should fail with constraintViolation
dn: uid=cjones,ou=People,dc=example,dc=com
changeType: modify
replace: employeeType
employeeType: fulltime
-

# this modify operation should succeed
dn: uid=cjones,ou=People,dc=example,dc=com
changeType: modify
replace: employeeType
employeeType: employee
-

# this modify operation should succeed
dn: uid=cjones,ou=People,dc=example,dc=com
changeType: modify
delete: employeeType
employeeType: employee
-
```

```
# this modify operation should fail with noSuchAttributeValue
dn: uid=cjones,ou=People,dc=example,dc=com
changeType: modify
delete: employeeType
employeeType: vice president
-

# this modify operation should fail with constraintViolation
dn: uid=cjones,ou=People,dc=example,dc=com
changeType: modify
add: employeeType
employeeType: employee
employeeType: vice president
-
```

Use the `ldapmodify` command one more time to verify that modify operations containing invalid `employeeType` values are rejected by the plug-in (and therefore by the directory server) and that those containing valid values are allowed. Listing 4.26 shows the command and the result. This time, the `-c` (continue if an error occurs) and the `-v` (verbose) options are used.

Listing 4.26 Testing the Value Constraint Plug-in with Modify Operations

```
./ldapmodify -c -v -D "uid=kvaughan,ou=People,dc=example,dc=com" -w bribery
↳ < testmodifies.ldif
ldapmodify: started Tue Aug  2 22:02:54 2002

ldap_init( localhost, 3389 )
Processing a version 1 LDIF file...
replace employeeType:
        fulltime
modifying entry uid=cjones,ou=People,dc=example,dc=com
ldap_modify: Constraint violation
ldap_modify: additional info: invalid value "fulltime" for employeeType

replace employeeType:
        employee
modifying entry uid=cjones,ou=People,dc=example,dc=com
modify complete

delete employeeType:
        employee
modifying entry uid=cjones,ou=People,dc=example,dc=com
modify complete

delete employeeType:
        vice president
modifying entry uid=cjones,ou=People,dc=example,dc=com
ldap_modify: No such attribute
```

```
add employeeType:
        employee
        vice president
modifying entry uid=cjones,ou=People,dc=example,dc=com
ldap_modify: Constraint violation
ldap_modify: additional info: invalid value "vice president" for employeeType
```

The plug-in worked as expected. Now perform one more quick test using a different LDAP client. Start Netscape Console and use the **Directory** tab to navigate to the **People** container. The `employeeType` attribute is not listed on any of Netscape's built-in tabs within the entry editor. Follow the steps described here to try to add an invalid `employeeType` value to an entry:

Step 1: Open an **Edit Entry** window for the person entry named `awhite` (Alan White) by double-clicking on the entry name in the right-hand pane of the **Directory** window. Alan White's entry is the first one listed in the **People** container.

Step 2: Click the **Advanced...** button to open a **Property Editor** window that provides access to all of the available attributes and object classes.

Step 3: Click the **Add Attribute...** button and select **employeeType** from the list that appears. An empty attribute labeled **Employee category** is added to the list of attributes in the **Property Editor** window. This is the descriptive name used by Netscape Console for the `employeeType` attribute.

Step 4: Type "Visitor" in the **Employee category** field, click the **OK** button to close the **Property Editor** window, and click the **OK** button on the **Edit Entry** window.

Netscape Console will try to write your changes to the directory server. Figure 4.11 shows the resulting error alert.

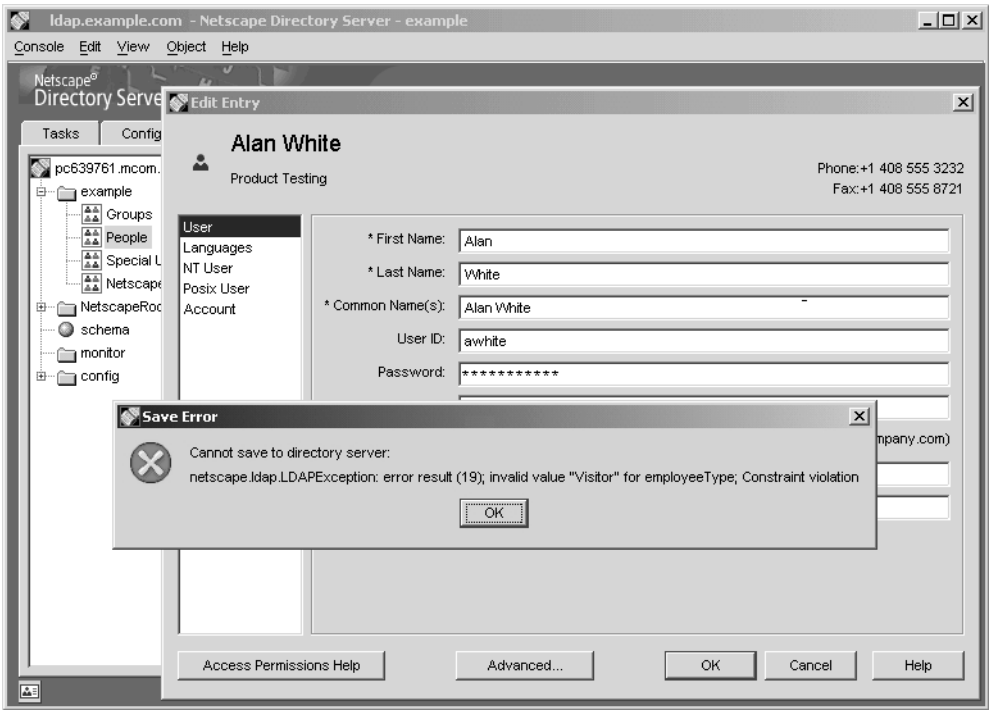
Although this was not an exhaustive test, you should now be convinced that the Value Constraint plug-in is working as designed. The ability to extend Netscape Directory Server using plug-ins is one of its strengths.

Ideas for Improvement

The Value Constraint plug-in could be enhanced in many ways. Here are a few ideas:

- Remove the hard-coded attribute name and the list of valid values from the `value-constraint.c` file, and change the plug-in to read the necessary configuration information from the Value Constraint plug-in's own entry.
- Support regular expressions or a similar, flexible pattern-matching scheme when checking for valid attribute values.

Figure 4.11 A Failed Attempt to Add an Invalid `employeeType` Value



- Modify the plug-in so that it enforces constraints for only those entries that contain a specific object class value. Although `employeeType` attributes typically appear only in person entries, you may want to enforce constraints on an attribute such as a person's `cn` (common name). Because the `cn` attribute is used in many other kinds of entries, in this case the Value Constraint plug-in needs to be intelligent enough to ignore add and modify operations for nonperson entries.

Your customized directory server is better able to meet your needs. As with a car that you own for many years, with experience you can make specific directory products perform in ways that better match your own needs. Now park the car and shut off the ignition:

```
./stop-slapd
```

This driving lesson is finished.

Further Reading

Definition of the inetOrgPerson LDAP Object Class (RFC 2798). M. Smith, 2000. Available on the Web at http://www.ietf.org/rfc/rfc2798.txt.

Netscape Directory Server Administrator's Guide. Available on the Web at http://enterprise.netscape.com/docs/directory.

Netscape Directory Server Configuration, Command, and File Reference. Available on the Web at http://enterprise.netscape.com/docs/directory.

Netscape Directory Server Deployment Guide. Available on the Web at http://enterprise.netscape.com/docs/directory.

Netscape Directory Server Installation Guide. Available on the Web at http://enterprise.netscape.com/docs/directory.

Netscape Directory Server Plug-in Programmer's Guide. Available on the Web at http://enterprise.netscape.com/docs/directory.

Netscape Directory Server Schema Reference. Available on the Web at http://enterprise.netscape.com/docs/directory.

Looking Ahead

Part I, Introduction to Directory Services and LDAP, has laid the groundwork for the rest of this book by providing an overview of directory services, a comprehensive introduction to the LDAP protocol, and a hands-on look at a leading directory server product. Part II, Designing Your Directory Service, will describe directory design from the ground up. Chapter 5, Directory Design Road Map, will provide an overview of the directory design process, and each of the remaining chapters of Part II will cover a major directory design topic in detail.