CHAPTER 12. SYNCHRONIZING RED HAT DIRECTORY SERVER WITH MICROSOFT ACTIVE DIRECTORY

Windows Sync carries over changes in a directory — adds, deletes, and changes in groups, users, and passwords — between Red Hat Directory Server and Microsoft Active Directory. This makes it much more efficient and effective to maintain consistent information across directories.

12.1. ABOUT WINDOWS SYNC

Synchronization allows the user and group entries in Active Directory to be matched with the entries in the Red Hat Directory Server. As entries are created, modified, or deleted, the corresponding change is made to the sync peer server, allowing two-way synchronization of users, passwords, and groups.

The synchronization process is analogous to the replication process: the synchronization is enabled by a plug-in, configured and initiated through a sync agreement, and record of directory changes is maintained and updates are sent according to that changelog. This synchronizes users and groups between Directory Server and a Windows server.

Windows Sync has two parts is configured in two parts, one for user and group entries and the other for passwords:

- *Directory Server Windows Sync.* Synchronization for user and group entries is configured in a synchronization agreement, much like replication is configured in a replication agreement. A sync agreement defines what kinds of entries are synchronized (users, groups, or both) and which direction changes are synced (from the Directory Server to Active Directory, from Active Directory to Directory Server, or both).

  The Directory Server relies on the Multi-Master Replication Plug-in to synchronize user and group entries. The same changelog that is used for multi-master replication is also used to send updates from the Directory Server to Active Directory as LDAP operations. The server also performs LDAP search operations against its Windows server to synchronize changes made to Windows entries to the corresponding Directory Server entry.

- *Password Sync Service.* Password changes made on Directory Server are automatically synced over to Active Directory, but there must be a special hook to recognize and transmit password changes on Active Directory over to Directory Server. This is done by the Password Sync Service. This application captures password changes on the Windows machines and send them to the Directory Server over LDAPS.

  The Password Sync Service must be installed on every Active Directory domain controller.

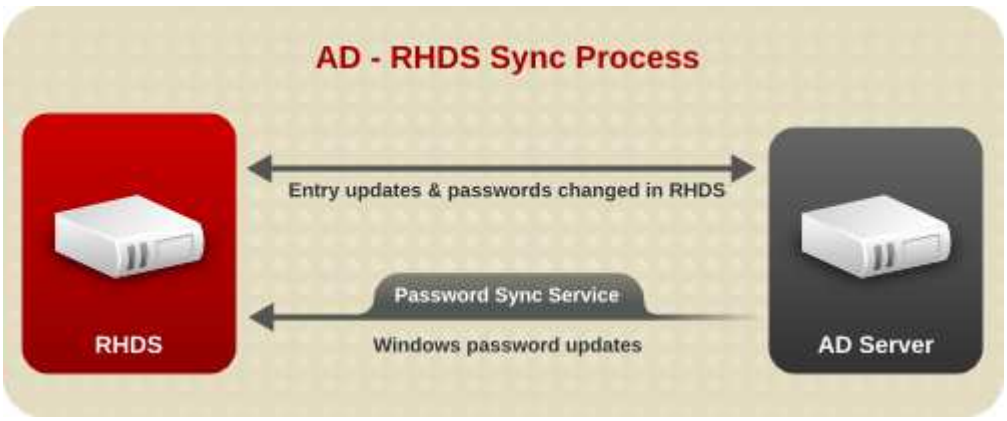


Figure 12.1. Active Directory — Directory Server Synchronization Process

Synchronization is configured and controlled by one or more *synchronization agreements*, which establishes synchronization between *sync peers*, the directory servers being synced. These are similar in purpose to replication agreements and contain a similar set of information, including the host name (or IPv4 or IPv6 address) and port number for Active Directory. The Directory Server connects to its peer Windows server using LDAP/LDAPS to both send and receive updates.

LDAP, a standard connection, can be used for syncing user and group entries alone, but to synchronize passwords, some sort of secure connection is required. If a secure connection is not used, the Windows domain will not accept password changes from the Directory Server and the Password Sync Service will not send passwords from the Active Directory domain to the Directory Server. Win Sync allows both LDAPS using SSL/TLS and Start TLS.

A single Active Directory subtree is synchronized with a single Directory Server subtree, and vice versa. Unlike replication, which connects *databases*, synchronization is between *suffixes*, parts of the directory tree structure. The synced Active Directory and Directory Server suffixes are both specified in the sync agreement. All entries within the

respective subtrees are candidates for synchronization, including entries that are not immediate children of the specified suffix DN.

NOTE

Any descendant container entries need to be created separately in Active Directory by an administrator; Windows Sync does not create container entries.

The Directory Server maintains a *changelog*, a database that records modifications that have occurred. The changelog is used by Windows Sync to coordinate and send changes made to the Active Directory peer. Changes to entries in Active Directory are found by using Active Directory's Dirsync search feature. The Dirsync search is issued periodically, every five minutes, to check for changes on the Active Directory server. Using Dirsync ensures that only those entries that have changed since the previous search are retrieved.

In some situations, such as when synchronization is configured or there have been major changes to directory data, a total update, or *resynchronization*, can be run. This examines every entry in both sync peers and sends any modifications or missing entries. A full Dirsync search is initiated whenever a total update is run. See Section 12.9, "Sending Synchronization Updates" for more information.

Windows Sync provides some control over which entries are synchronized to grant administrators fine-grained control of the entries that are synchronized and to give sufficient flexibility to support different deployment scenarios. This control is set through different configuration attributes set in the Directory Server:

- When creating the sync agreement, there is an option to synchronizing new Windows entries (***nsDS7NewWinUserSyncEnabled*** and ***nsDS7NewWinGroupSyncEnabled***) as they are created. If these attributes are set to **on**, then existing Windows users/groups are synchronized to the Directory Server, and users/groups as they are created are synchronized to the Directory Server.

  Within the Windows subtree, only entries with user or group object classes can be synchronized to Directory Server.

- On the Directory Server, only entries with the **ntUser** or **ntGroup** object classes and attributes can be synchronized.

The placement of the sync agreement depends on what suffixes are synchronized; for a single suffix, the sync agreement is made for that suffix alone; for multiple suffixes, the sync agreement is made at a higher branch of the directory tree. To propagate Windows entries and updates throughout the Directory Server deployment, make the agreement between a master in a multi-master replication environment, and use that master to replicate the changes across the Directory Server deployment, as shown in Figure 12.2, "Multi-Master Directory Server – Windows Domain Synchronization".

IMPORTANT

While it is possible to configure a sync agreement on a hub server, this only allows uni-directional synchronization, from Red Hat Directory Server to Active Directory. The Active Directory server cannot sync any changes back to the hub.

It is strongly recommended that only masters in multi-master replication be used to configure synchronization agreements.

WARNING

There can only be a single sync agreement between the Directory Server environment and the Active Directory environment. Multiple sync agreements to the same Active Directory domain can create entry conflicts.

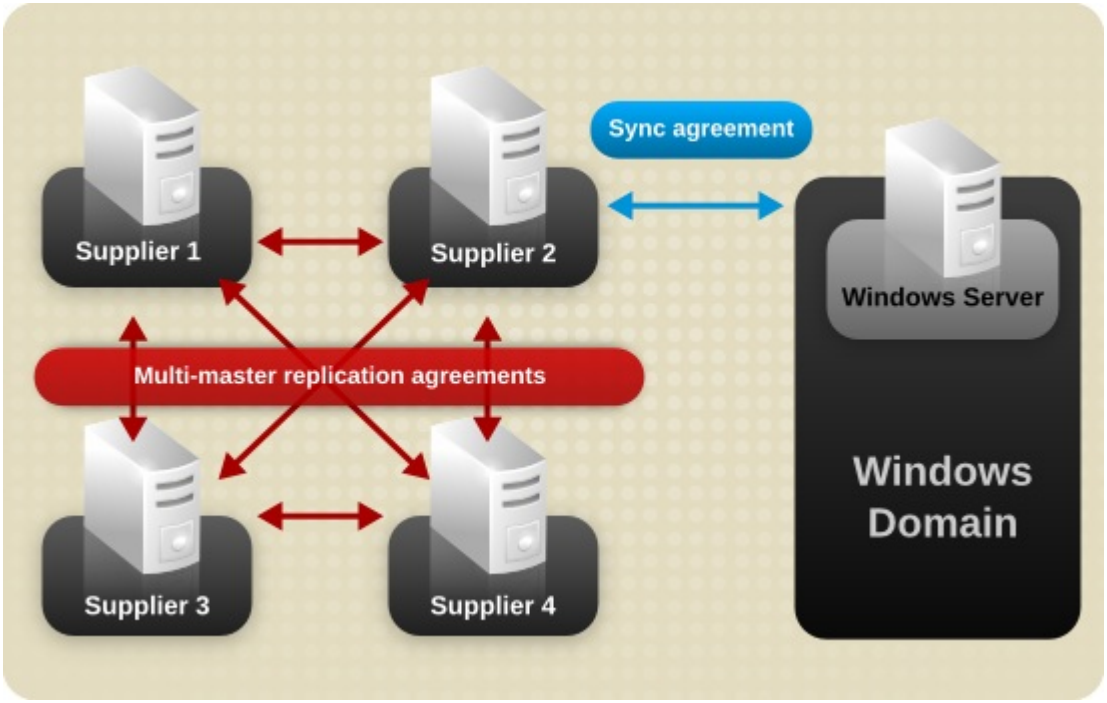


Figure 12.2. Multi-Master Directory Server — Windows Domain Synchronization

Directory Server passwords are synchronized along with other entry attributes because plain-text passwords are retained in the Directory Server changelog. The Password Sync service is needed to catch password changes made on Active Directory. Without the Password Sync service, it would be impossible to have Windows passwords synchronized because passwords are hashed in Active Directory, and the Windows hashing function is incompatible with the one used by Directory Server.

12.2. SUPPORTED ACTIVE DIRECTORY VERSIONS

Windows Synchronization and the Password Sync Service are supported on Windows 2008 R2 on both 32-bit and 64-bit platforms.

12.3. STEPS FOR CONFIGURING WINDOWS SYNC

Configuring synchronization is very similar to configuring replication. It requires configuring the database as a master with a changelog and creating an agreement to define synchronization. A common user identity, a sync user, connects to the Windows sync peer to send updates from the Directory Server and to check for updates to sync back to the Directory Server.

NOTE

To synchronize passwords (which is the only way for users to be active on both Directory Server and Active Directory), synchronization must be configured to run over SSL/TLS. Therefore, this configuration section assumes that SSL/TLS must also be configured.

Configuring synchronization over SSL/TLS is also similar to configuring replication over SSL/TLS. Both sync peers must be configured to trust each other for encrypted sessions (all password operations are performed over TLS/SSL).

All synchronization for user and group entries is passive from the Active Directory side; it is the Directory Server which sends updates on its side and polls for updates on the Active Directory domain. For passwords, the Active Directory server requires a separate password service; this service actively sends password changes from the Active Directory domain to Directory Server.

12.3.1. Step 1: Configure SSL on Directory Server

The full instructions for configuring the Directory Server to run in SSL are at Section 7.4.2, "Enabling TLS/SSL Only in the Directory Server". Basically, the Directory Server needs to have the appropriate SSL certificates installed, be configured to run over an SSL port, and allow client authentication from other servers.

Two certificates must be issued and installed on both the Directory Server and the Active Directory sync peer:

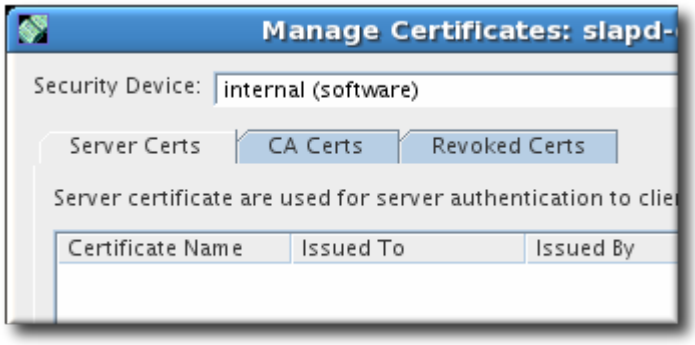
- CA certificate, shared between the Directory Server and Active Directory

- Server certificates for the Directory Server and Active Directory sync peers, which are accessible by the sync services

To set up SSL:

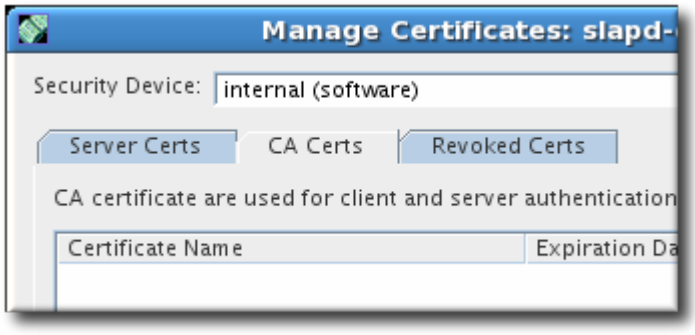
1. Generate a certificate request.
    1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.

    2. Select the **Server Certs** tab, and click the **Request** button at the bottom.



    3. Fill in the certificate information, and save the certificate request to a file.
2. Submit the certificate to a certificate authority, and retrieve it once it is issued.

    The method for submitting certificate requests and retrieving certificates varies for each CA.
3. Install the new certificate.
    1. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
    2. Select the **Server Certs** tab, and click **Install** at the bottom of the window.
    3. Paste in the certificate, and set the password for the token database.
4. Install the CA certificate for the issuing CA.
    1. Download and save the CA certificate from the CA's site. Each CA has a slightly different way of making its CA certificate available.
    2. In the Directory Server Console, select the **Tasks** tab, and click **Manage Certificates**.
    3. Go to the **CA Certs** tab, and click **Install** at the bottom of the window.

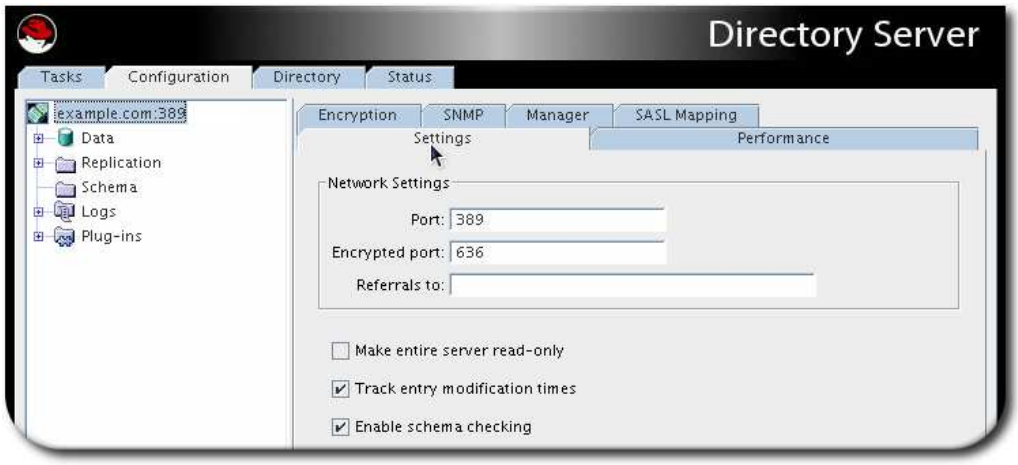


    4. Paste in the CA certificate or point to the downloaded file, and go through the certificate installer.

5. Change the server to the SSL port; this is described in much more detail in Section 1.6, "Changing Directory Server Port Numbers".

    1. Open the Directory Server Console, and open the **Configuration** tab for the Directory Server.

    2. In the **Settings** tab, set the secure port for the server to use for TLS/SSL communications, such as **636**. Click **Save**.



    3. Select the **Encryption** tab in the right pane.

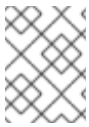
    4. Select the **Enable SSL for this Server** check box, then select the certificate to use from the drop-down menu. Click **Save**.

5. Restart the Directory Server. The Directory Server must be restarted from the command line.

```
service dirsrv restart example
```

To restart the Directory Server without the password prompt, create a PIN file or use a hardware crypto device. See Section 7.4.4, "Creating a Password File for the Directory Server" for information on how to create a PIN file.

12.3.2. Step 2: Configure the Active Directory Domain

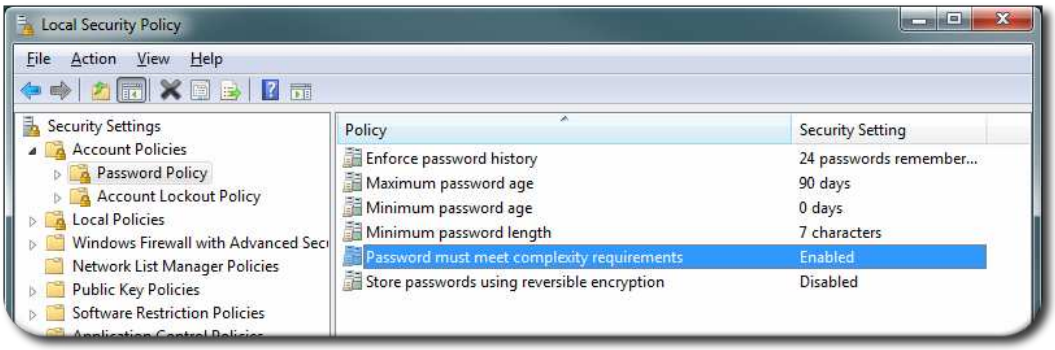


NOTE

Synchronization can only be configured with an Active Directory domain controller, so make sure that the domain is properly installed and configured.

The first configuration step is to make sure that the Active Directory password complexity policies are are enabled so that the Password Sync service will run.

1. Run **secpol.msc** from the command line.
2. Select **Security Settings**.
3. Open **Account Policies**, and then open **Password Policy**.
4. Enable the **Password must meet complexity requirements** option and save.



Configure SSL and set up a root CA on the Active Directory server, as described in the Microsoft knowledgebase at http://technet.microsoft.com/en-us/library/cc772393%28v=ws.10%29.aspx#BKMK_AS1.

1. Install a certificate authority.
    1. In the **Administrative Tools** area, open **Server Manager** and add a role.
    2. Select the **Active Directory Certificate Services** check box.
    3. Click through to the **Select Role Services** page, and select the **Certification Authority** check box.
    4. When configuring the CA, select the following options on the appropriate screens:
        - **Enterprise** for the setup type

- **Certification Authority Web Enrollment** in the optional configuration

5. Reboot the Active Directory server.

2. Set up the Active Directory server to use the SSL server certificate.

    1. Create a certificate request **.inf**, using the fully-qualified domain name of the Active Directory as the certificate subject. For example:

```
;----------------- request.inf -----------------

[Version]

Signature="$Windows NT$

[NewRequest]

Subject = "CN=ad.server.example.com, O=Engineering, L=Raleigh, S=North Carolina, C=US"
KeySpec = 1
KeyLength = 2048
Exportable = TRUE
MachineKeySet = TRUE
SMIME = False
PrivateKeyArchive = FALSE
UserProtected = FALSE
UseExistingKeySet = FALSE
ProviderName = "Microsoft RSA SChannel Cryptographic Provider"
ProviderType = 12
RequestType = PKCS10
KeyUsage = 0xa0

[EnhancedKeyUsageExtension]

OID=1.3.6.1.5.5.7.3.1

;-------------------------------------------------
```

    For more information on the **.inf** request file, see the Microsoft documentation, such as http://technet.microsoft.com/en-us/library/cc783835.aspx.

    2. Generate the certificate request.

```
certreq -new request.inf request.req
```

    3. Submit the request to the Active Directory CA. For example:

```
certreq -submit request.req certnew.cer
```

NOTE

If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is **http:**//*servername*/**certsrv**.

    4. Accept the certificate request. For example:

```
certreq -accept certnew.cer
```

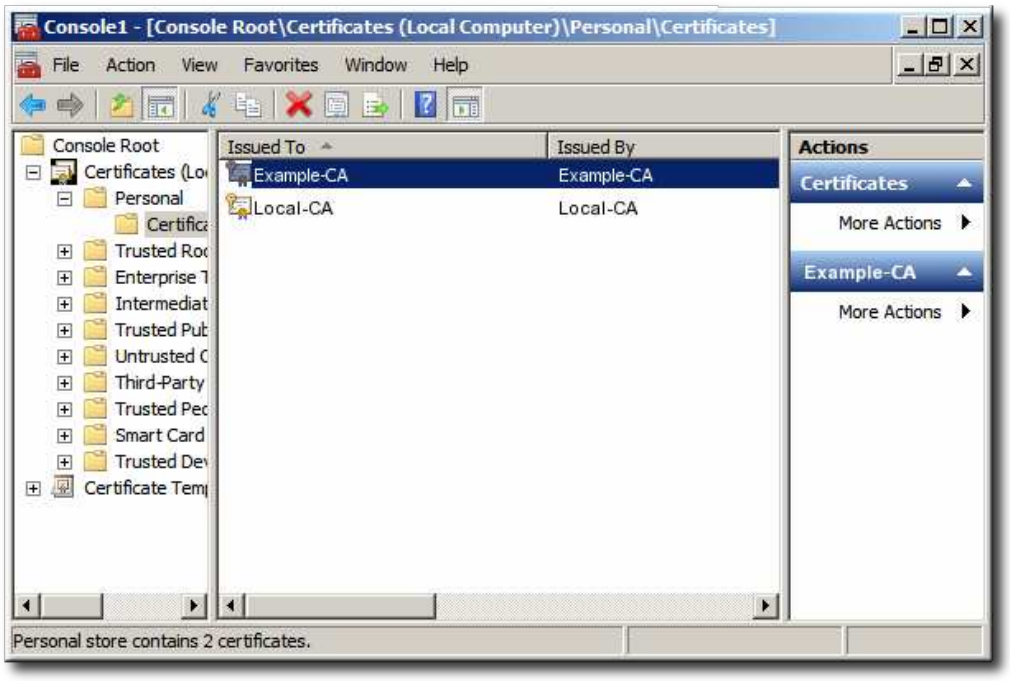
3. Make sure that the server certificate is present on the Active Directory server.

    1. In the **Run** menu, open the MMC console.

    2. In the **File** menu, click **Add/Remove Snap-in...**.

    3. Select the **Certificates** snap-in, and click **Add** to add it, and then click **Next**.

    4. Expand the **Certificates (Local)** menu on the left. Expand the **Personal** item and click **Certificates**.



5. The new certificate should be listed with the other certificates.

4. On the Directory Server, export the CA certificate.

```
[root@server ~]# cd /etc/dirsrv/slapd-instance_name
[root@server slapd-instance_name]# certutil -d . -L -n "CA certificate" -a > dsca.crt
```

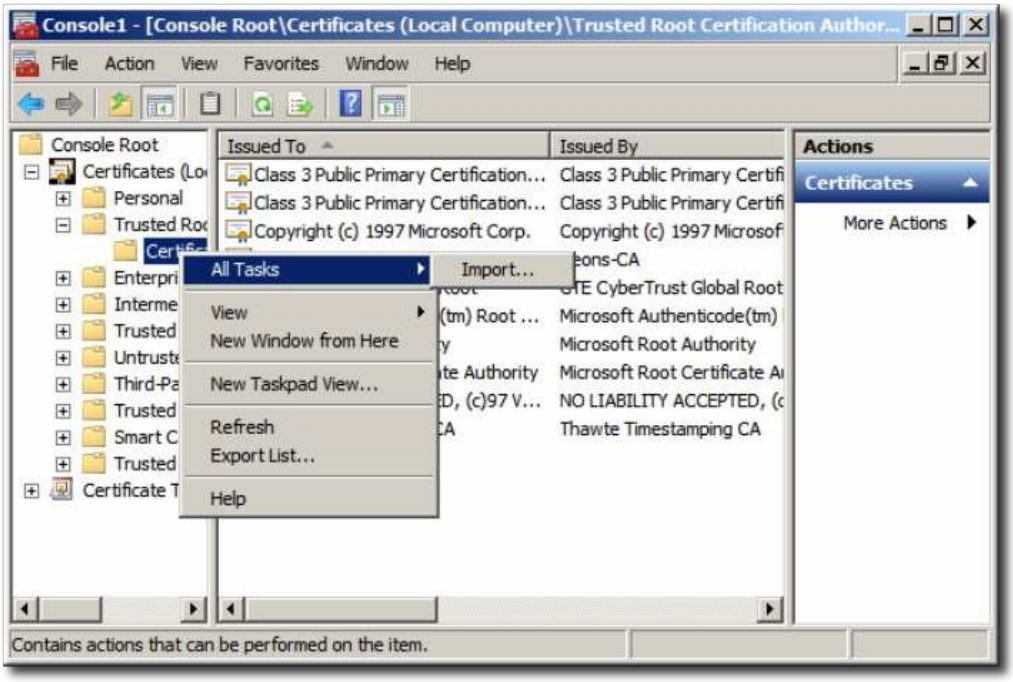
5. Copy the exported certificate from the Directory Server to the Windows machine.

6. Import the CA certificate from Directory Server into Active Directory.

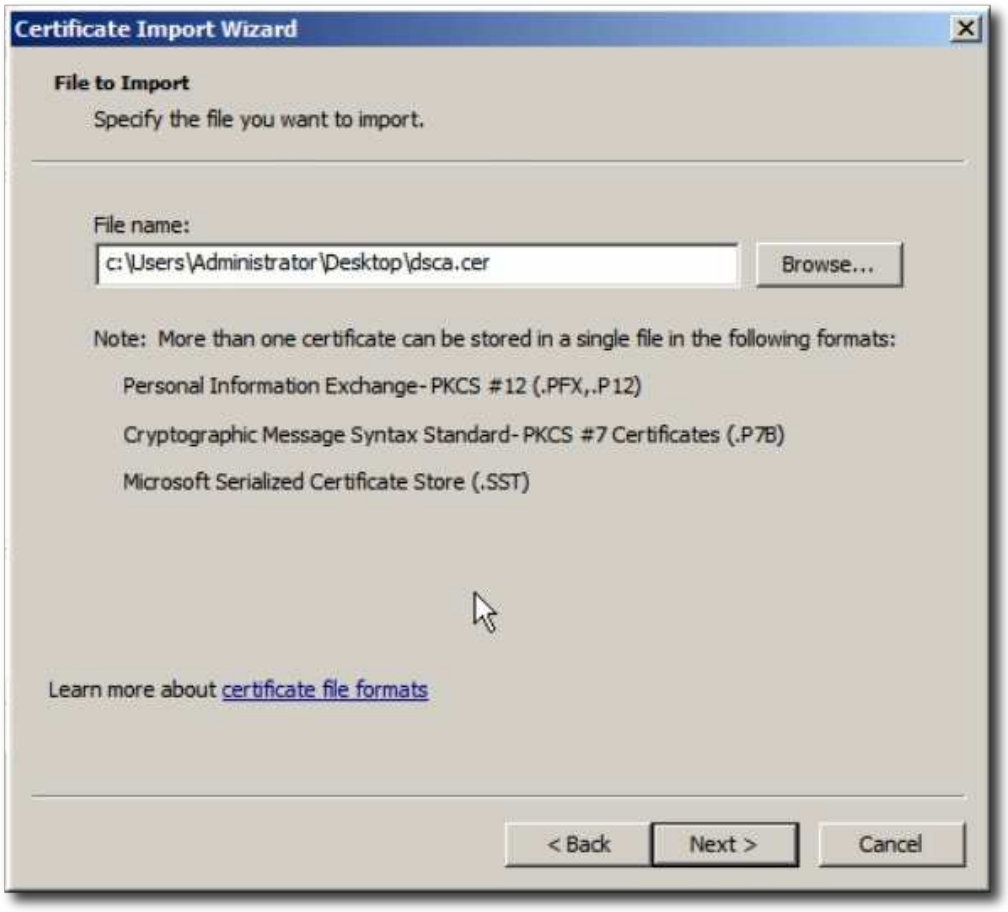
    1. Open **Administrative Tools** and select the **Certificate Authority** item.

    2. Expand **Trusted Root Certification Authorities**.

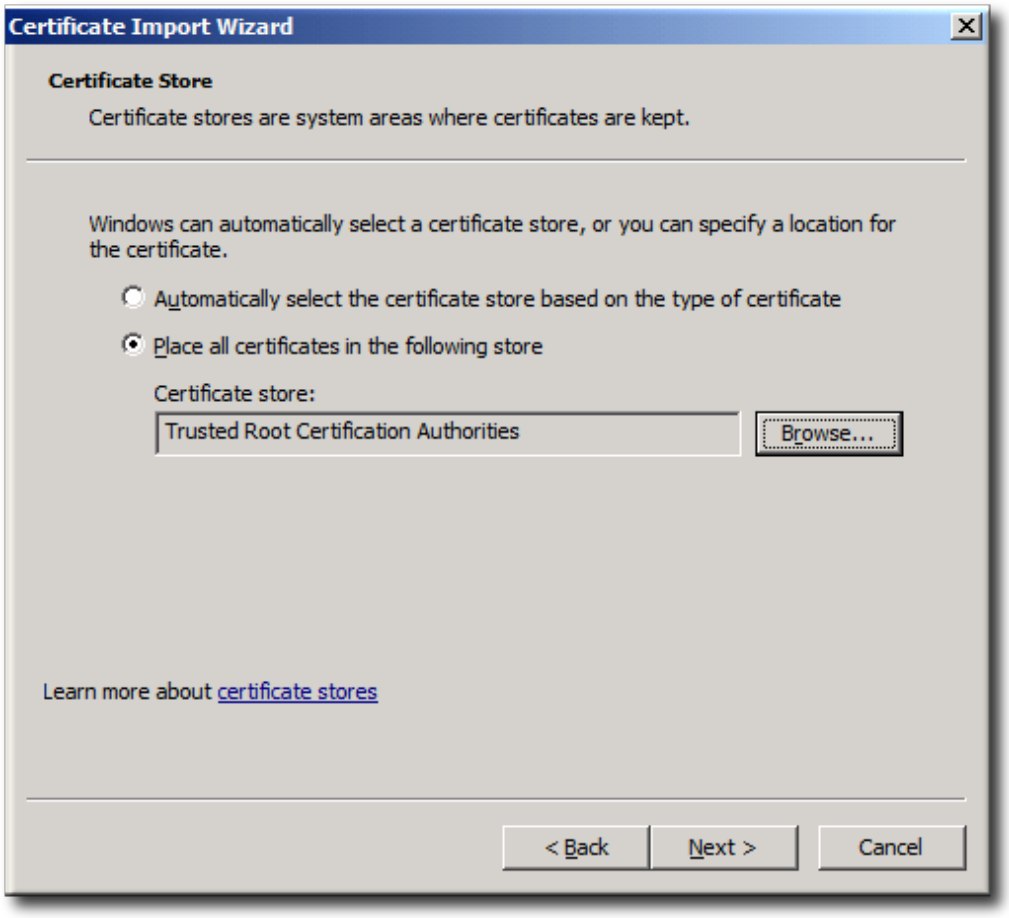
3. Right-click the **Certificates** item and select **Import**.



4. Browse to the downloaded Directory Server CA certificate, and click **Next**.



5. Save the CA certificate in the **Trusted Root Certification Authorities** store.



7. Reboot the domain controller.

To test that the server is running in SSL correctly, try searching Active Directory over LDAPS.

12.3.3. Step 3: Select or Create the Sync Identity

There are two users used to configure Windows Sync:

- *An Active Directory user, specified in the sync agreement.*

  The user specified in the sync agreement is the entity as whom the Directory Server binds to Active Directory to send and receive updates. The Active Directory user should be a member of the Domain Admins group, or have equivalent rights, and must have rights to replicate directory changes.

  For information on adding users and setting privileges in Active Directory, see the Microsoft documentation.

- *A Directory Server user, specified in the Password Sync Service.*

  The user referenced in the Password Sync Service must have read and write permissions to every entry within the synchronized subtree and absolutely must have write access to password attributes in Directory Server so that Password Sync can update password changes.

NOTE

The user cited in the sync agreement (the supplier DN) exists on the Active Directory server. The user cited in the Password Sync configuration exists on Directory Server.

To create a sync user on Directory Server:

1. Create a new entry, such as **cn=sync user,cn=config**, with a password. For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=sync user,cn=config
changetype: add
objectClass: inetorgperson
objectClass: person
```

```
objectClass: top
cn: sync user
sn: SU
userPassword: secret
passwordExpirationTime: 20380119031407Z
```

2. Set an ACI that grants the sync user access to compare and write user passwords.

   The ACI must be set at the top of the subtree which will be synchronized. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: ou=people,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="userPassword")(version 3.0;acl "password sync";allow (write,compare)
userdn="ldap:///cn=sync user,cn=config";)
```

   For security reasons, the Password Sync user should not be Directory Manager and should not be part of the synchronized subtree.

12.3.4. Step 4: Install the Password Sync Service

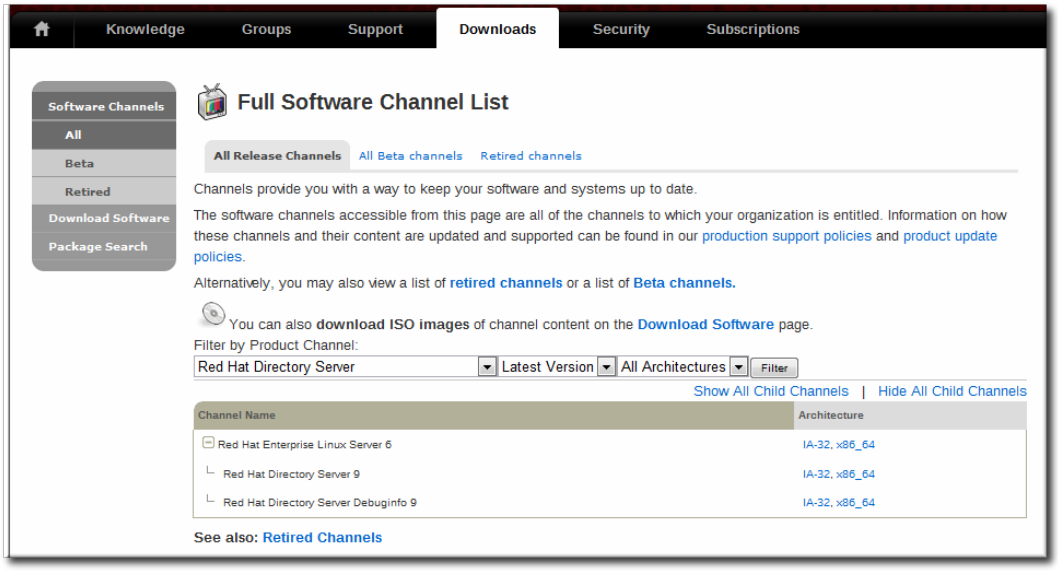
IMPORTANT

Password Sync must be installed on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

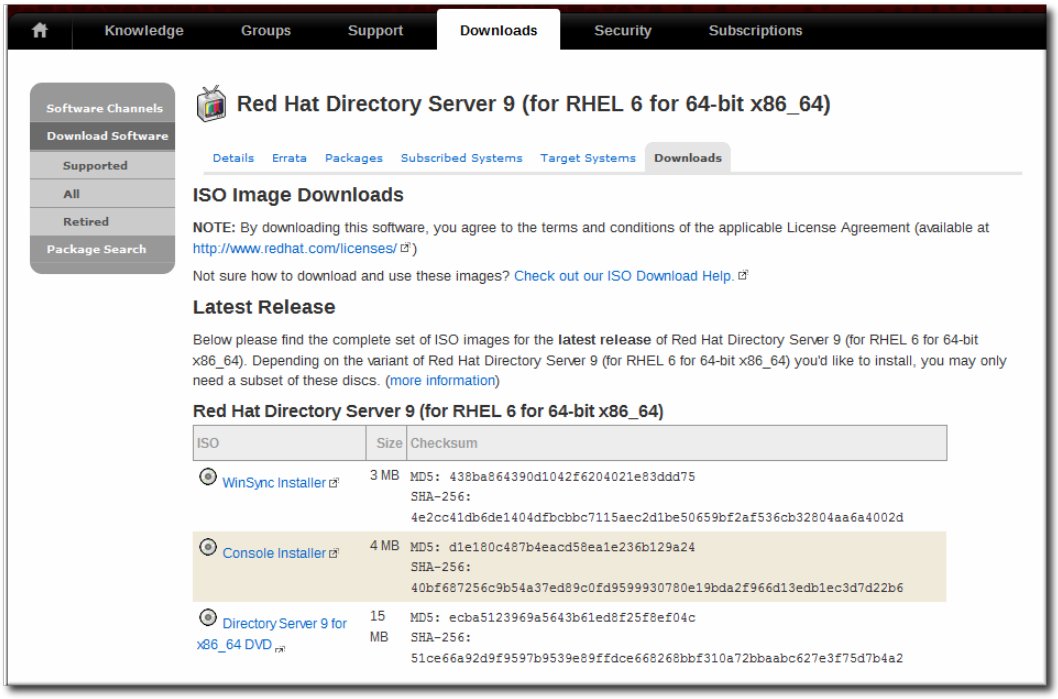
Passwords can only be synchronized if both the Directory Server and Windows server are running in SSL, the sync agreement is configured over an SSL connection, and certificate databases are configured for Password Sync to access.

The Password Sync Service is supported on Microsoft Windows Server 2008 R2 (32-bit and 64-bit).

1. Go to http://access.redhat.com.
2. Click the **Downloads** tab, and select the Red Hat Enterprise Linux channels, then filter for the Directory Server product and architecture.



3. Open the **Downloads** tab for the Directory Server channel.



4. On the Directory Server page, download the appropriate version of the WinSync Installer. This is the Password Sync MSI file (**RedHat-PassSync-1.1.5-*arch*.msi**). Save it to the Active Directory machine.

NOTE

There are two PassSync packages available, one for 32-bit Windows servers and one for 64-bit. Make sure to select the appropriate packages for your Windows platform.

5. Double-click the Pass Sync MSI file to install it.

6. The **Password Sync Setup** window appears. Hit **Next** to begin installing.

7. Fill in the Directory Server host name (or IPv4 or IPv6 address), secure port number, user name (such as **cn=sync user,cn=config**), the certificate token (password), and the search base (e.g., **ou=People,dc=example,dc=com**).

Hit **Next**, then **Finish** to install Password Sync.

8. Reboot the Windows machine to start Password Sync.

NOTE

The Windows machine must be rebooted. Without the rebooting, **PasswordHook.dll** is not enabled, and password synchronization will not function.

The first attempt to synchronize passwords, which happened when the Password Sync application is installed, will always fail because of the SSL connection between the Directory Server and Active Directory sync peers. The tools to create the certificate and key databases is installed with the **.msi**.

Password Sync and many of its libraries are installed in **C:\Program Files\Red Hat Directory Password Synchronization**. All of the files installed with Password Sync are listed in Table 12.1, "Installed Password Sync Libraries".

Table 12.1. Installed Password Sync Libraries

Directory	Library	Directory	Library
C:\WINDOWS\system32	passhook.dll	C:\WINDOWS\system32	libnspr4.dll
C:\WINDOWS\system32	nss3.dll	C:\WINDOWS\system32	sqlite3.dll
C:\WINDOWS\system32	softokn3.dll	C:\WINDOWS\system32	nssdbm3.dll
C:\WINDOWS\system32	nssutil3.dll		
C:\WINDOWS\system32	smime3.dll	C:\WINDOWS\system32	freebl3.dll
C:\Program Files\Red Hat Directory Password Synchronization	nsldap32v60.dll	C:\Program Files\Red Hat Directory Password Synchronization	certutil.exe

Directory	Library	Directory	Library
C:\Program Files\Red Hat Directory Password Synchronization	nsldappr32v60.dll	C:\Program Files\Red Hat Directory Password Synchronization	nsldapssl32v60.dll
C:\WINDOWS\system32	ssl3.dll	C:\WINDOWS\system32	libplc4.dll
C:\Program Files\Red Hat Directory Password Synchronization	nssckbi.dll	C:\Program Files\Red Hat Directory Password Synchronization	nsldif32v60.dll
C:\Program Files\Red Hat Directory Password Synchronization	passsync.log[a]	C:\Program Files\Red Hat Directory Password Synchronization	passsync.exe
C:\Program Files\Red Hat Directory Password Synchronization	pk12util.exe	C:\Program Files\Red Hat Directory Password Synchronization	msvcr71.dll
C:\WINDOWS\system32	libplds4.dll		

[a] This log file is not an installed library, but it is created at installation.

12.3.5. Step 5: Configure the Password Sync Service

Next, set up certificates that Password Sync uses to access the Directory Server over SSL:

NOTE

SSL is required for Password Sync to send passwords to Directory Server. The service will not send the passwords except over SSL to protect the clear text password sent from the Active Directory machine to the Directory Server machine. This means that Password Sync will not work until SSL is configured.

1. On the Directory Server, export the CA certificate.

```
[root@server ~]# cd /etc/dirsrv/slapd-instance_name
[root@server ~]# certutil -d . -L -n "CA certificate" -a > dsca.crt
```

2. Copy the exported certificate from the Directory Server to the Windows machine.
3. Open a command prompt on the Windows machine, and open the **Password Sync** installation directory.

```
C:\Windows\system32>cd "C:\Program Files\Red Hat Directory Password Synchronization"
```

4. Create new **cert8.db** and **key.db** databases on the Windows machine.

```
C:\C:\Program Files\Red Hat Directory Password Synchronization>certutil.exe -d . -N
```

5. Import the CA certificate from the Directory Server into the new certificate database.

```
certutil.exe -d . -A -n "DS CA cert" -t CT,, -a -i \path\to\dsca.crt
```

6. Verify that the CA certificate was correctly imported.

```
certutil.exe -d . -L -n "DS CA cert"
```

7. Reboot the Windows machine. The Password Sync service is not available until after a system reboot.

NOTE

If any Active Directory user accounts exist when Password Sync is first installed, then the passwords for those user accounts cannot be synchronized until they are changed because Password Sync cannot decrypt a password once it has been hashed in Active Directory.

12.3.6. Step 6: Configure the Directory Server Database for Synchronization

Just as with replication, there must be a changelog available to track and send directory changes and the Directory Server database being synchronized must be configured as a replica.

NOTE

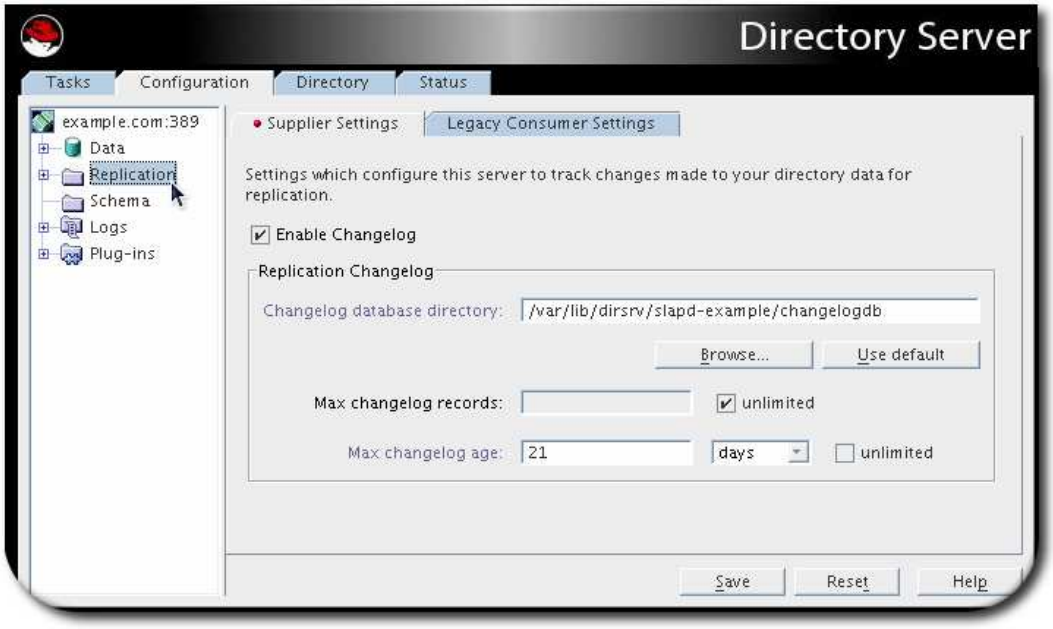
If the Directory Server database is already configured for replication, this step is not necessary.

Setting up a database for replication is described in Section 11.5.1, "Configuring the Read-Write Replicas on the Supplier Servers".

12.3.6.1. Setting up the Directory Server from the Console

First, enable the changelog:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click the **Replication** folder.
3. In the main window, click the **Supplier Settings** tab.
4. Check the **Enable Changelog** database.



5. Set the changelog database directory. Click the **Use default** button to use the default or **Browse...** to select a custom directory.
6. Save the changelog settings.

After setting up the changelog, then configure the database that will be synchronized as a replica. The replica role should be either a single-master or multi-master.

IMPORTANT

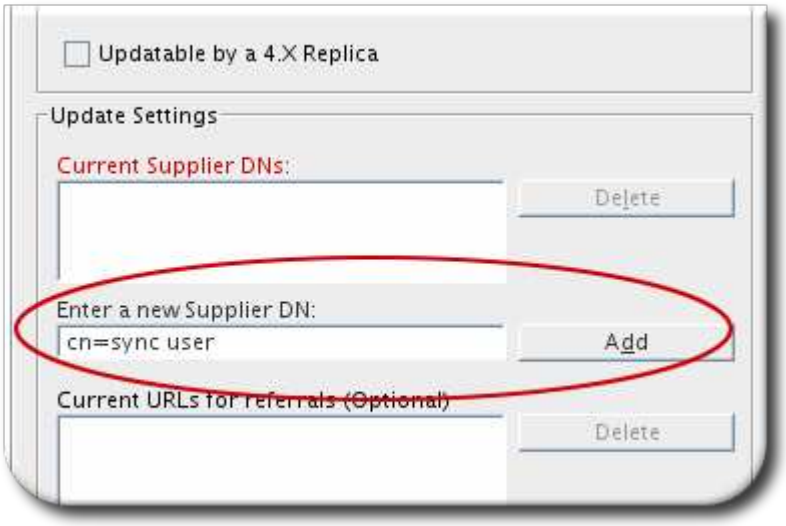
While it is possible to configure a sync agreement on a hub server, this only allows uni-directional synchronization, from Red Hat Directory Server to Active Directory. The Active Directory server cannot sync any changes back to the hub.

It is strongly recommended that only masters in multi-master replication be used to configure synchronization agreements.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click the **Replication** folder, then click the name of the database to synchronize.

   By default, there are two databases, **NetscapeRoot** for directory configuration and **userRoot** for directory entries. Other databases may be listed if they have been added to Directory Server.
3. Check the **Enable Replica** check box, and select the radio button by the type of replica which the database is.

4. In the **Update Settings** section, either select or add a supplier DN. This is the user account as which synchronization process will be run. As mentioned in Section 12.3.3, "Step 3: Select or Create the Sync Identity", this user must be on the Active Directory server.



5. Save the replication settings for the database.

NOTE

For more information on replication settings, see Chapter 11, *Managing Replication*.

12.3.6.2. Setting up the Directory Server for Sync from the Command Line

First, enable the changelog:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb
```

Then, create the supplier replica entry:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: sync replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 7
nsds5replicatype: 3
nsds5flags: 1
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=sync user,cn=config
```

These different parameters are described in more detail in the *Configuration and Command-Line Tool Reference* and Section 11.7.1, "Configuring Suppliers from the Command Line" .

12.3.7. Step 7: Create the Synchronization Agreement

Create the synchronization agreement.

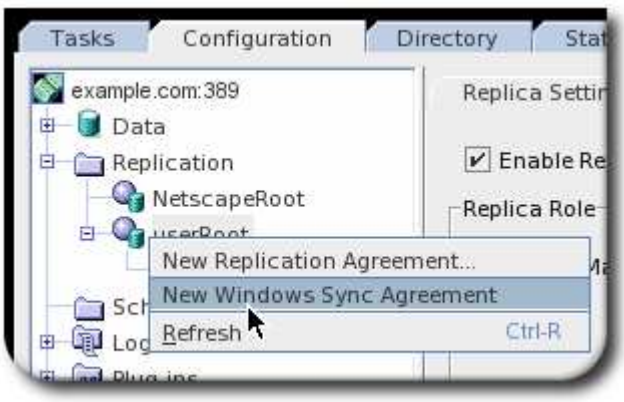
NOTE

If secure binds are required for simple password authentication (Section 14.8.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (SSL/TLS and Start TLS connections or SASL authentication) is recommended, anyway.

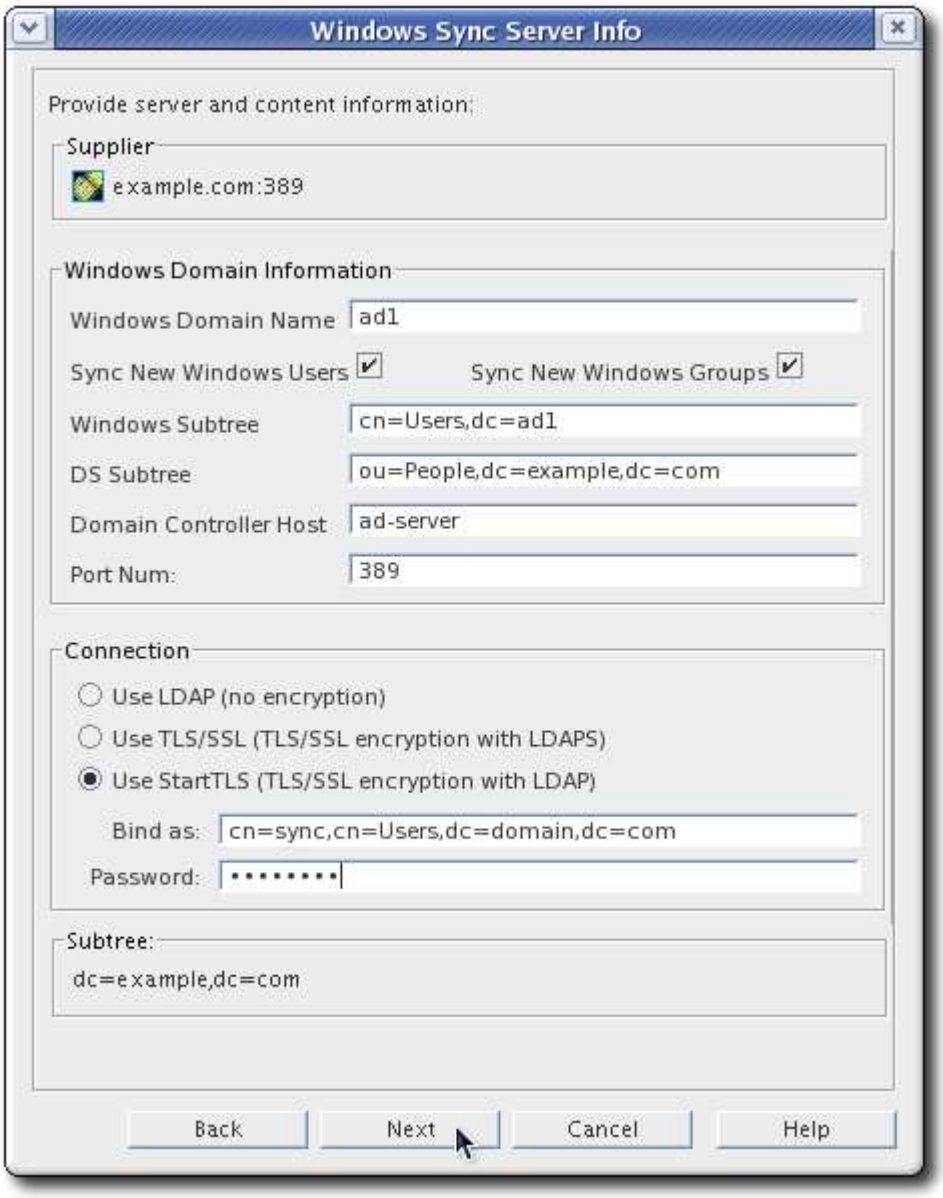
12.3.7.1. Creating the Sync Agreement from the Console

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click **Replication**, then right-click on the database to sync. The default user database is **userRoot**, but additional databases are added as new suffixes are added to the Directory Server.

   Alternatively, highlight the database, and in the top tool bar, click **Object**.
3. Select **New Windows Sync Agreement** from the menu.



4. In the two fields, supply a name and description of the synchronization agreement. Hit **Next**.
5. In the **Windows Sync Server Info** window, fill in the Active Directory information in the **Windows Domain Information** area.



- The name of the Windows domain.
- What kinds of entries to synchronize; users and groups are synchronized independently. When a type of entry is chosen, then all of the entries of that type that are found in the Windows subtree are created in the Directory Server.
- The Windows and Directory Server subtree information; this is automatically filled in.
- The host name, IPv4 address, or IPv6 address of the domain controller
- The Windows server's port number

6. Set the connection type. There are three options:
   - *Use LDAP*. This sets either a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. Both the Directory Server and the Windows server must be properly configured to run in TLS/SSL for this connection and must have installed each other's CA certificates in order to trust their server certificates.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port. Like regular SSL, these peer servers must be able to trust each other's certificates.

   Using either TLS/SSL or Start TLS is recommended for security reasons. TLS/SSL or Start TLS is required for synchronizing passwords because Active Directory refuses to modify passwords unless the connection is SSL-protected.

7. Fill in the authentication information in the **Bind as...** and **Password** fields with the sync ID information. This user must exist in the Active Directory domain.

8. Save the sync agreement.

NOTE

By default, Win Sync polls the Active Directory peer every five (5) minutes to check for changes. In the sync agreement summary, this is displayed as the **Update Interval**. The update interval can be changed by editing the ***winSyncInterval*** attribute manually. See Section 12.10.2, "Adding and Editing the Sync Agreement in the Command Line".

When the agreement is complete, the new sync agreement is listed under the suffix.

12.3.7.2. Creating the Sync Agreement from the Command Line

It is also possible to add the sync agreement through the command line.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsDSWindowsReplicationAgreement
cn: ExampleSyncAgreement
nsds7WindowsReplicaSubtree: cn=Users,dc=ad1
nsds7DirectoryReplicaSubtree: ou=People,dc=example,dc=com
nsds7NewWinUserSyncEnabled: on
nsds7NewWinGroupSyncEnabled: on
nsds7WindowsDomain: ad1
nsDS5ReplicaRoot: dc=example,dc=com
nsDS5ReplicaHost: ad1.windows-server.com
nsDS5ReplicaPort: 389
nsDS5ReplicaBindDN: cn=sync user,cn=config
nsDS5ReplicaCredentials: {DES}ffGad646dT0nnsT8nJOaMA==
nsDS5ReplicaTransportInfo: TLS
winSyncInterval: 1200
```

All of the different parameters used in the sync agreement are listed in Table 12.6, "Sync Agreement Attributes". These different parameters are described in more detail in the *Configuration and Command-Line Tool Reference* .

12.3.8. Step 8: Configure Directory Server User and Group Entries for Synchronization

Add the **ntUser** and **ntGroup** object classes to any user and group entries, respectively, which will be synchronized, along with any required attributes. Only Directory Server entries with those object classes are synchronized. Active Directory entries which are synced over to Directory Server have those object classes automatically.

Whenever the appropriate object classes are added to an entry, both for new entries and existing entries, the entry is synced over at the next incremental update.

Configuring Directory Server user entries for synchronization is described in Section 12.4.3, "Configuring User Sync for Directory Server Users", and configuring Directory Server group entries for synchronization is described in Section 12.5.4, "Configuring Group Sync for Directory Server Groups" .

12.3.9. Step 9: Begin Synchronization

After the sync agreement is created, begin the synchronization process.

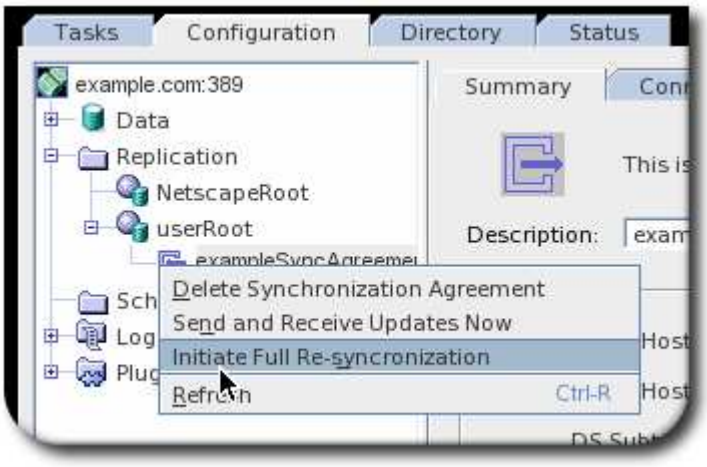
1. Go to the **Configuration** tab in the Console.

2. Open the **Replication** folder and expand the appropriate database.

3. Select the sync agreement.

4. Right-click on the agreement or open the **Object** menu.

5. Select **Initiate Full Re-synchronization**.



If synchronization stops for any reason, begin another total update (resynchronization) by selecting this from the sync agreement menu. Beginning a total update (resynchronization) will not delete or overwrite the databases.

12.4. SYNCHRONIZING USERS

Users are not automatically synced between Directory Server and Active Directory. Synchronization both directions has to be configured:

- Users in the Active Directory domain are synced if it is configured in the sync agreement by selecting the **Sync New Windows Users** option. All of the Windows users are copied to the Directory Server when synchronization is initiated and then new users are synced over when they are created.
- A Directory Server user account is synchronized to Active Directory through specific attributes that are present on the Directory Server entry. Any Directory Server entry must have the **ntUser** object class and the ***ntUserCreateNewAccount*** attribute; the ***ntUserCreateNewAccount*** attribute (even on an existing entry) signals Directory Server Win Syn to write the entry over to the Active Directory server.

  New or modified user entries with the **ntUser** object class added are created and synced over to the Windows machine at the next regular update, which is a standard poll of entry.

NOTE

A user is not active on the Active Directory domain until it has a password. When an existing user is modified to have the required Windows attributes, that user entry will be synced over to the Active Directory domain, but will not be able to log in until the password is changed on the Directory Server side or an administrator sets the password on Active Directory. This is because passwords stored in the Directory Server are encrypted, and Password Sync cannot sync already encrypted passwords.

To make the user active on the Active Directory domain, reset the user's password.

All synchronized entries *in the Directory Server*, whether they originated in the Directory Server or in Active Directory, have special synchronization attributes:

- *ntUserDomainId.* This corresponds to the ***sAMAccountName*** attribute for Active Directory entries.
- *ntUniqueId.* This contains the value of the ***objectGUID*** attribute for the corresponding Windows entry. This attribute is set by the synchronization process and should not be set or modified manually.
- *ntUserDeleteAccount.* This attribute is set automatically when a Windows entry is synced over but must be set manually for Directory Server entries. If ***ntUserDeleteAccount*** has the value **true**, the corresponding Windows entry be deleted when the Directory Server entry is deleted. Otherwise, the entry remains in Active Directory, but is removed from the Directory Server database if it is deleted in the Directory Server.

Setting ***ntUserCreateNewAccount*** and ***ntUserDeleteAccount*** on Directory Server entries allows the Directory Manager precise control over which users within the synchronized subtree are synced on Active Directory.

12.4.1. User Attributes Synchronized between Directory Server and Active Directory

Only a subset of Directory Server and Active Directory attributes are synchronized. These attributes are hard-coded and are defined regardless of which way the entry is being synchronized. Any other attributes present in the entry, either in Directory Server or in Active Directory, remain unaffected by synchronization.

Some attributes used in Directory Server and Active Directory are identical. These are usually attributes defined in an LDAP standard, which are common among all LDAP services. These attributes are synchronized to one another exactly. Table 12.3, "User Schema That Are the Same in Directory Server and Windows Servers" shows attributes that are the same between the Directory Server and Windows servers.

Some attributes define the same information, but the names of the attributes or their schema definitions are different. These attributes are mapped between Active Directory and Directory Server, so that attribute A in one server is treated as attribute B in the other. For synchronization, many of these attributes relate to Windows-specific information. Table 12.2, "User Schema Mapped between Directory Server and Active Directory" shows the attributes that are mapped between the Directory Server and Windows servers.

For more information on the differences in ways that Directory Server and Active Directory handle some schema elements, see Section 12.4.2, "User Schema Differences between Red Hat Directory Server and Active Directory" .

Table 12.2. User Schema Mapped between Directory Server and Active Directory

Directory Server	Active Directory
cn[a]	name
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

[a] The ***cn*** is treated differently than other synced attributes. It is mapped directly (***cn*** to ***cn***) when syncing from Directory Server to Active Directory. When syncing from Active Directory to Directory Server, however, ***cn*** is mapped from the ***name*** attribute on Windows to the ***cn*** attribute in Directory Server.

Table 12.3. User Schema That Are the Same in Directory Server and Windows Servers

cn[a]	physicalDeliveryOfficeName
description	postOfficeBox
destinationIndicator	postalAddress
facsimileTelephoneNumber	postalCode
givenname	registeredAddress
homePhone	sn

homePostalAddress	st
initials	street
l	telephoneNumber
mail	teletexTerminalIdentifier
mobile	telexNumber
o	title
ou	usercertificate
pager	x121Address

[a] The ***cn*** is treated differently than other synced attributes. It is mapped directly (***cn*** to ***cn***) when syncing from Directory Server to Active Directory. When syncing from Active Directory to Directory Server, however, ***cn*** is mapped from the ***name*** attribute on Windows to the ***cn*** attribute in Directory Server.

12.4.2. User Schema Differences between Red Hat Directory Server and Active Directory

Although Active Directory supports the same basic X.500 object classes as Directory Server, there are a few incompatibilities of which administrators should be aware.

12.4.2.1. Values for cn Attributes

In Directory Server, the ***cn*** attribute can be multi-valued, while in Active Directory this attribute must have only a single value. When the Directory Server ***cn*** attribute is synchronized, then, only one value is sent to the Active Directory peer.

What this means for synchronization is that,potentially, if a ***cn*** value is added to an Active Directory entry and that value is not one of the values for ***cn*** in Directory Server, then all of the Directory Server ***cn*** values are overwritten with the single Active Directory value.

One other important difference is that Active Directory uses the ***cn*** attribute attribute as its naming attribute, where Directory Server uses ***uid***. This means that there is the potential to rename the entry entirely (and accidentally) if the ***cn*** attribute is edited in the Directory Server. If that ***cn*** change is written over to the Active Directory entry, then the entry is renamed, and the new named entry is written back over to Directory Server.

12.4.2.2. Password Policies

Both Active Directory and Directory Server can enforce password policies such as password minimum length or maximum age. Windows Sync makes no attempt to ensure that the policies are consistent, enforced, or synchronized. If password policy is not consistent in both Directory Server and Active Directory, then password changes made on one system may fail when synced to the other system. The default password syntax setting on Directory Server mimics the default password complexity rules that Active Directory enforces.

12.4.2.3. Values for street and streetAddress

Active Directory uses the attribute ***streetAddress*** for a user or group's postal address; this is the way that Directory Server uses the ***street*** attribute. There are two important differences in the way that Active Directory and Directory Server use the ***streetAddress*** and ***street*** attributes, respectively:

- In Directory Server, ***streetAddress*** is an alias for ***street***. Active Directory also has the ***street*** attribute, but it is a separate attribute that can hold an independent value, not an alias for ***streetAddress***.
- Active Directory defines both ***streetAddress*** and ***street*** as single-valued attributes, while Directory Server defines ***street*** as a multi-valued attribute, as specified in RFC 4519.

Because of the different ways that Directory Server and Active Directory handle ***streetAddress*** and ***street*** attributes, there are two rules to follow when setting address attributes in Active Directory and Directory Server:

- Windows Sync maps ***streetAddress*** in the Windows entry to ***street*** in Directory Server. To avoid conflicts, the ***street*** attribute should not be used in Active Directory.

- Only one Directory Server ***street*** attribute value is synced to Active Directory. If the ***streetAddress*** attribute is changed in Active Directory and the new value does not already exist in Directory Server, then all ***street*** attribute values in Directory Server are replaced with the new, single Active Directory value.

12.4.2.4. Constraints on the initials Attribute

For the ***initials*** attribute, Active Directory imposes a maximum length constraint of six characters, but Directory Server does not have a length limit. If an ***initials*** attribute longer than six characters is added to Directory Server, the value is trimmed when it is synchronized with the Active Directory entry.

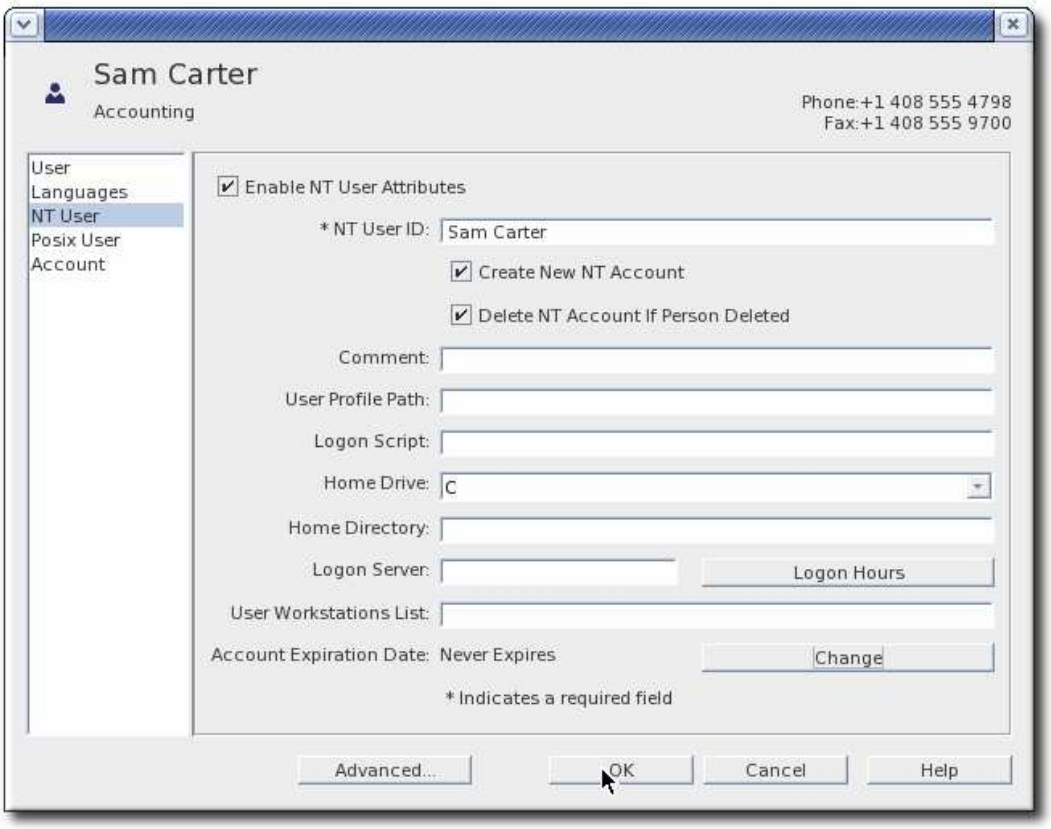
12.4.3. Configuring User Sync for Directory Server Users

For Directory Server users to be synchronized over to Active Directory, the user entries must have the appropriate sync attributes set.

12.4.3.1. Configuring User Sync in the Console

1. In the Directory Server Console, select the **Directory** tab.
2. For an existing entry, right-click the entry, and click **Properties** to open the property editor for the entry.

   For a new entry, right-click the main entry in the left window to add the new entry, select **User**, and then fill in the required entry attributes.
3. On the left side of the **Property Editor**, click the **NT User** link.
4. In the **NT User** tab, check the **Enable NT Attributes** check box.



5. To enable synchronization, two fields are required:
   - Setting a **NT User ID**
   - Selecting the **Create New NT Account** check box
6. Selecting the **Delete NT Account** check box means that the corresponding Windows user is deleted if the Directory Server entry is deleted.
7. Set the other Windows attributes. These attributes are mapped to relevant Windows attributes.

Additional **ntUser** attributes can be created either by using the **Advanced** button; see Section 3.2.4.2, "Modifying Entries Using ldapmodify".

NOTE

Reset the user's password.

A user is not active on the Active Directory domain until it has a password. When an existing user is modified to have the required Windows attributes, that user entry will be synced over to the Active Directory domain, but will not be able to log in until the password is changed on the Directory Server side or an administrator sets the password on Active Directory. Password Sync cannot sync encrypted passwords.

So, to make the user active on the Active Directory domain, reset the user's password.

12.4.3.2. Configuring User Sync in the Command Line

To enable synchronization through the command line, add the required sync attributes to an entry or create an entry with those attributes.

Three schema elements are required for synchronization:

- The **ntUser** object class
- The ***ntUserDomainId*** attribute, to give the Windows ID
- The ***ntUserCreateNewAccount*** attribute, to signal to the synchronization plug-in to sync the Directory Server entry over to Active Directory

For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: uid=scarter,ou=People,dc=example,dc=com
changetype: modify
add: objectClass
objectClass:ntUser

add: ntUserDomainId
ntUserDomainId: Sam Carter

add: ntUserCreateNewAccount
ntUserCreateNewAccount: true

add: ntUserDeleteAccount
ntUserDeleteAccount: true
```

Many additional Windows and user attributes can be added to the entry. All of the schema which is synchronized is listed in Section 12.4.1, "User Attributes Synchronized between Directory Server and Active Directory" . Windows-specific attributes, belonging to the **ntUser** object class, are described in more detail in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*.

NOTE

Reset the user's password.

A user is not active on the Active Directory domain until it has a password. When an existing user is modified to have the required Windows attributes, that user entry will be synced over to the Active Directory domain, but will not be able to log in until the password is changed on the Directory Server side or an administrator sets the password on Active Directory. Password Sync cannot sync encrypted passwords.

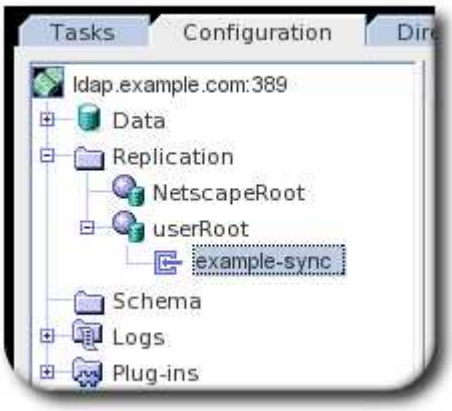
So, to make the user active on the Active Directory domain, reset the user's password.

12.4.4. Configuring User Sync for Active Directory Users

Synchronization for Windows users (users which originate in the Active Directory domain) is configured in the sync agreement.

12.4.4.1. Configuring User Sync in the Console

1. Open the **Configuration** tab and expand the **Replication** folder.
2. Open the appropriate database, and select the sync agreement.



3. Open the **Connection** tab.
4. Check the **New Windows User Sync** check box to enable users sync. To disable sync, uncheck the box.

For new sync agreements, select the corresponding users sync check box in the sync agreement creation wizard.

12.4.4.2. Configuring User Sync in the Command Line

The attribute to set Active Directory user sync is ***nsds7NewWinUserSyncEnabled*** and is set on the sync agreement. To enable user sync, add this attribute to the sync agreement or create a sync agreement with this attribute set to **on**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=userRoot,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
replace: nsds7NewWinUserSyncEnabled
nsds7NewWinUserSyncEnabled: on
```

To disable user sync, set **nsds7NewWinUserSyncEnabled: off**.

12.5. SYNCHRONIZING GROUPS

Like user entries, groups are not automatically synced between Directory Server and Active Directory. Synchronization both directions has to be configured:

- Groups in the Active Directory domain are synced if it is configured in the sync agreement by selecting the **Sync New Windows Groups** option. All of the Windows groups are copied to the Directory Server when synchronization is initiated and then new groups are synced over as they are created.
- A Directory Server group account is synchronized to Active Directory through specific attributes that are present on the Directory Server entry. Any Directory Server entry must have the **ntGroup** object class and the ***ntGroupCreateNewGroup*** attribute; the ***ntGroupCreateNewGroup*** attribute (even on an existing entry) signals Directory Server Win Sync to write the entry over to the Active Directory server.

  New or modified groups that have the **ntGroup** object class are created and synced over to the Windows machine at the next regular update.

IMPORTANT

When a group is synced, the list of all of its members is also synced. However, the member entries themselves are not synced unless user sync is enabled and applies to those entries.

This could created a problem is an application or service tries to do a modify operation on all members in a group on the Active Directory server, and some of those users do not exist.

Additionally, groups have a few other common attributes:

- Two attributes control whether Directory Server groups are created and deleted on Active Directory, ***ntGroupCreateNewGroup*** and ***ntGroupDeleteGroup***.

  ntGroupCreateNewGroup is required to sync Directory Server groups over to Active Directory.

- *ntUserDomainId* contains the unique ID for the entry on the Active Directory domain. This is the only required attribute for the **ntGroup** object class.

- *ntGroupType* is the type of Windows group. Windows group types are global/security, domain local/security, global/distribution, or domain local/distribution. This is set automatically for Windows groups that are synchronized over, but this attribute must be set manually on Directory Server entries before they can be synced.

12.5.1. About Windows Group Types

In Active Directory, there are two major types of groups: security and distribution. Security groups are most similar to groups in Directory Server, since security groups can have policies configured for access controls, resource restrictions, and other permissions. Distribution groups are for mailing distribution. These are further broken down into global and local groups. The Directory Server *ntGroupType* supports all four group types:

- **-21483646** for global/security (the default)
- **-21483644** for domain local/security
- **2** for global/distribution
- **4** for domain local/distribution

12.5.2. Group Attributes Synchronized between Directory Server and Active Directory

Only a subset of Directory Server and Active Directory attributes are synchronized. These attributes are hard-coded and are defined regardless of which way the entry is being synchronized. Any other attributes present in the entry, either in Directory Server or in Active Directory, remain unaffected by synchronization.

Some attributes used in Directory Server and Active Directory group entries are identical. These are usually attributes defined in an LDAP standard, which are common among all LDAP services. These attributes are synchronized to one another exactly. Table 12.5, "Group Entry Attributes That Are the Same between Directory Server and Active Directory" shows attributes that are the same between the Directory Server and Windows servers.

Some attributes define the same information, but the names of the attributes or their schema definitions are different. These attributes are mapped between Active Directory and Directory Server, so that attribute A in one server is treated as attribute B in the other. For synchronization, many of these attributes relate to Windows-specific information. Table 12.4, "Group Entry Attribute Mapping between Directory Server and Active Directory" shows the attributes that are mapped between the Directory Server and Windows servers.

For more information on the differences in ways that Directory Server and Active Directory handle some schema elements, see Section 12.5.3, "Group Schema Differences between Red Hat Directory Server and Active Directory" .

Table 12.4. Group Entry Attribute Mapping between Directory Server and Active Directory

Directory Server	Active Directory
cn	name
ntUserDomainID	name
ntGroupType	groupType

Directory Server	Active Directory
uniqueMember member	Member[a]

[a] The ***Member*** attribute in Active Directory is synced to the ***uniqueMember*** attribute in Directory Server.

Table 12.5. Group Entry Attributes That Are the Same between Directory Server and Active Directory

cn	o
description	ou
l	seeAlso
mail	

12.5.3. Group Schema Differences between Red Hat Directory Server and Active Directory

Although Active Directory supports the same basic X.500 object classes as Directory Server, there are a few incompatibilities of which administrators should be aware.

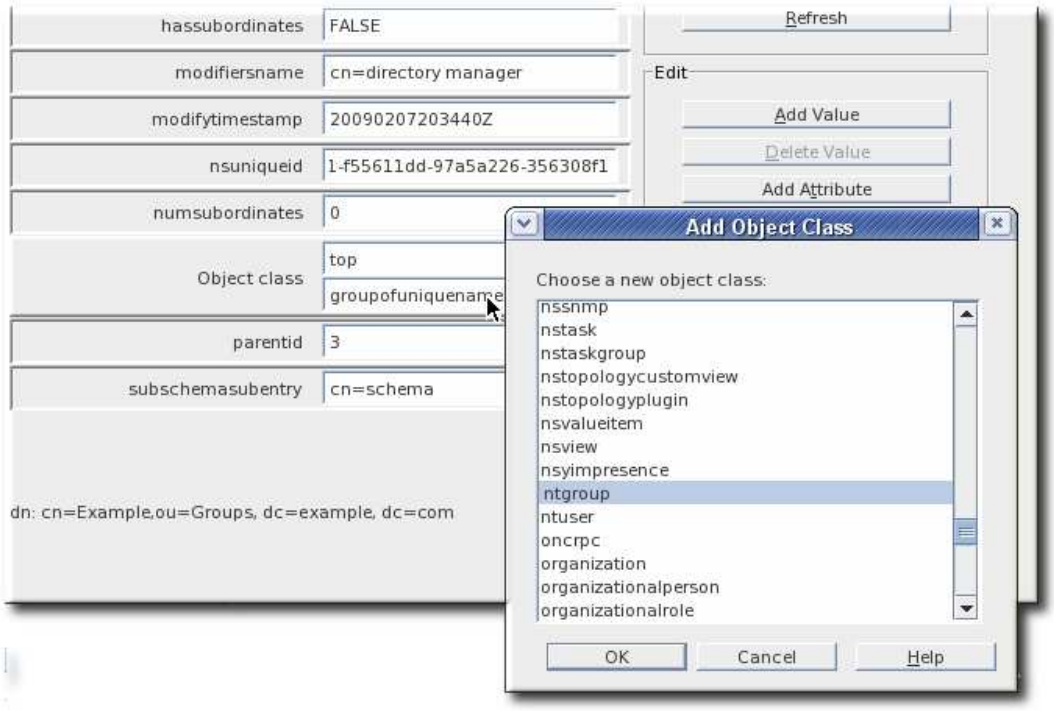
Nested groups (where a group contains another group as a member) are supported and for WinSync are synchronized. However, Active Directory imposes certain constraints as to the composition of nested groups. For example, a global group contain a domain local group as a member. Directory Server has no concept of local and global groups, and, therefore, it is possible to create entries on the Directory Server side that violate Active Directory's constraints when synchronized.

12.5.4. Configuring Group Sync for Directory Server Groups

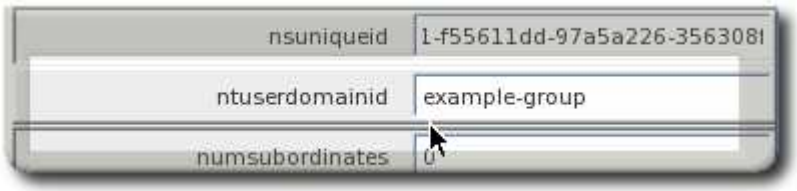
For Directory Server groups to be synchronized over to Active Directory, the group entries must have the appropriate sync attributes set.

12.5.4.1. Configuring Group Sync in the Console

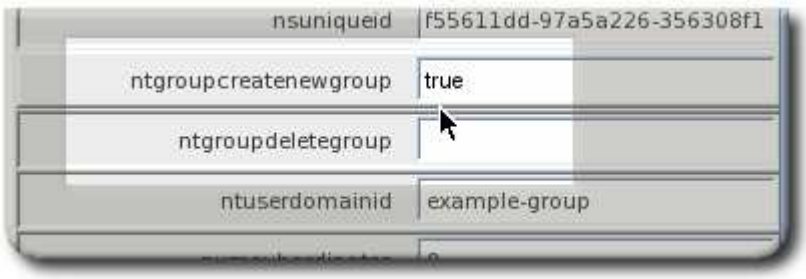
1. In the Directory Server Console, select the **Directory** tab.
2. Right-click the group entry, and click **Advanced** to open the advanced property editor for the entry. All of the sync-related attributes must be added manually, so only the advanced property editor can set the attributes.
3. Click the **objectClasses** field, and then click the **Add Value** button.
4. Select the **ntGroup** object class.



5. Setting the **ntGroup** object class automatically adds the ***ntUserDomainId*** attribute. This attribute is required, so add a value.



6. To enable synchronization, click the **Add Attribute** button, and select the ***ntGroupCreateNewGroup*** attribute from the list. Then, set its value to **true**. This signals to the sync plug-in that the entry should be added to the Active Directory directory.



To delete the group entry from the Active Directory domain if it is deleted from the Directory Server database, set the ***ntGroupDeleteGroup*** attribute and set it to **true**.

7. Add any other Windows attributes for the Directory Server entry. The available attributes are listed in Section 12.5.2, "Group Attributes Synchronized between Directory Server and Active Directory" .

If the ***ntGroupType*** is not added, then the group is automatically added as a global security group (**ntGroupType:-21483646**).

12.5.4.2. Configuring Group Sync in the Command Line

To enable synchronization through the command line, add the required sync attributes to an entry or create an entry with those attributes.

Three schema elements are required for synchronization:

- The **ntGroup** object class.

- The ***ntUserDomainId*** attribute, to give the Windows ID for the entry.
- The ***ntGroupCreateNewGroup*** attribute, to signal to the synchronization plug-in to sync the Directory Server entry over to Active Directory.

  The ***ntGroupDeleteGroup*** attribute is optional, but this sets whether to delete the entry automatically from the Active Directory domain if it is deleted in the Directory Server.

It is also recommended to the ***ntGroupType*** attribute. If this attribute is not specified, then the group is automatically added as a global security group (**ntGroupType:-21483646**).

For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Example Group,ou=Groups,dc=example,dc=com
changetype: modify
add: objectClass
objectClass:ntGroup

add: ntUserDomainId
ntUserDomainId: example-group

add: ntGroupCreateNewGroup
ntGroupCreateNewGroup: true

add: ntGroupDeleteGroup
ntGroupDeleteGroup: true

add: ntGroupType
ntGroupType: 2
```

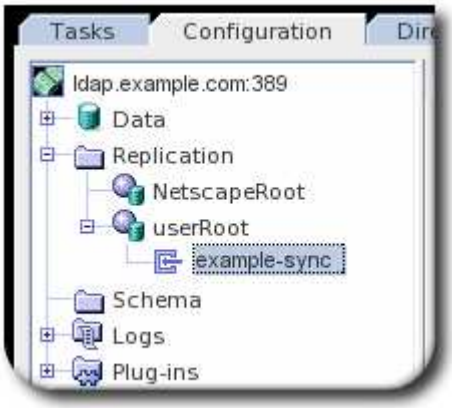
Many additional Windows and group attributes can be added to the entry. All of the schema which is synchronized is listed in Section 12.5.2, "Group Attributes Synchronized between Directory Server and Active Directory" . Windows-specific attributes, belonging to the **ntGroup** object class, are described in more detail in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*.

12.5.5. Configuring Group Sync for Active Directory Groups

Synchronization for Windows users (users which originate in the Active Directory domain) is configured in the sync agreement.

12.5.5.1. Configuring Group Sync in the Console

1. Open the **Configuration** tab and expand the **Replication** folder.
2. Open the appropriate database, and select the sync agreement.



3. Open the **Connection** tab.
4. Check the **New Windows Group Sync** check box to enable group sync. To disable sync, uncheck the box.

For new sync agreements, select the corresponding group sync check box in the sync agreement creation wizard.

12.5.5.2. Configuring Group Sync in the Command Line

The attribute to set Active Directory group sync is ***nsds7NewWinGroupSyncEnabled*** and is set on the sync agreement. To enable group sync, add this attribute to the sync agreement or create a sync agreement with this attribute set to **on**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=userRoot,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
replace: nsds7NewWinGroupSyncEnabled
nsds7NewWinGroupSyncEnabled: on
```

To disable group sync, set **nsds7NewWinGroupSyncEnabled: off**.

12.6. CONFIGURING UNI-DIRECTIONAL SYNCHRONIZATION

As Figure 12.1, "Active Directory — Directory Server Synchronization Process" illustrates, synchronization is *bi-directional* by default. That means that changes in Active Directory are sent to Directory Server and changes on Directory Server are sent to Active Directory.

It is possible to create *uni-directional* synchronization, where changes are only sent one-way. This is similar to a master-consumer relationship[3] as opposed to multi-master.

An additional attribute for the sync agreement, ***oneWaySync***, enables uni-directional synchronization and specifies the direction to send changes. The possible values are ***fromWindows*** (for Active Directory to Directory Server sync) and ***toWindows*** (for Directory Server to Active Directory sync). If this attribute is absent, then synchronization is bi-directional.

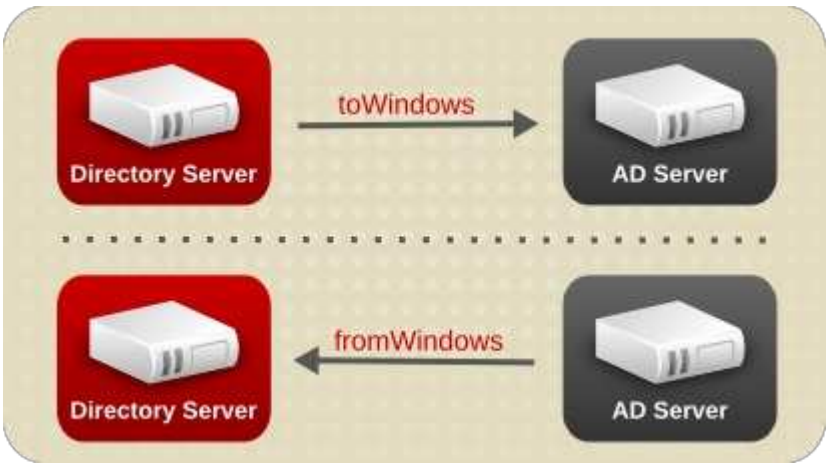


Figure 12.3. Uni-Directional Synchronization

The synchronization process itself is the mostly same for bi-directional and uni-directional synchronization. It uses the same sync interval and configuration. The only difference is in how sync information is requested.

For Windows-only sync, during the regular synchronization update interval, the Directory Server contacts the Active Directory server and sends the DirSync control to request updates. However, the Directory Server does not send any changes or entries from its side. So, the sync update consists of the Active Directory changes being sent to and updating the Directory Server entries.

For Directory Server only sync, the Directory Server sends entry modifications to the Active Directory server in a normal update, but it does not include the DirSync control so that it does not request any updates from the Active Directory side.

To enable uni-directional sync:

1. Create the synchronization agreement, as in Section 12.3.7, “Step 7: Create the Synchronization Agreement” .
2. There is no option in the Directory Server Console to set uni-directional sync when the agreement is initially created. Edit the sync agreement to contain the ***oneWaySync*** attribute.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
add: oneWaySync
oneWaySync: fromWindows
```

NOTE

Enabling uni-directional sync does *not* automatically prevent changes on the un-synchronized server, and this can lead to inconsistencies between the sync peers between sync updates. For example, uni-directional sync is configured to go from Active Directory to Directory Server, so Active Directory is (in essence) the data master. If an entry is modified or even deleted on the Directory Server, then the Directory Server information is different then the information and those changes are never carried over to Active Directory. During the next sync update, the edits are overwritten on the Directory Server and the deleted entry is re-added.

To prevent data inconsistency, use access control rules to prevent editing or deleting entries within the synchronized subtree on the *un*synced server. Access controls for Directory Server are covered in Section 13.3, “Creating ACIs Manually” . For Active Directory, see the appropriate Windows documentation.

12.7. SYNCHRONIZING POSIX ATTRIBUTES FOR USERS AND GROUPS

A subset of all possible user and attributes are synchronized between Active Directory and Red Hat Directory Server. Some attributes are mapped, where there are differences between Active Directory and Directory Server schemas, and some attributes are matched directly. The attributes (matched and mapped) which are synchronized are listed in Section 12.4.1, “User Attributes Synchronized between Directory Server and Active Directory” and Section 12.5.2, “Group Attributes Synchronized between Directory Server and Active Directory”.

By default, only those attributes are synchronized.

One type of attribute that is missing from that sync list is any POSIX-related attribute. On Linux systems, system users and groups are identified as POSIX entries, and LDAP POSIX attributes contain that required information. However, when Windows users are synced over, they have ***ntUser*** and ***ntGroup*** attributes automatically added which identify them as Windows accounts, but no POSIX attributes are synced over (even if they exist on the Active Directory entry) and no POSIX attributes are added on the Directory Server side.

The Posix Winsync API Plug-in synchronizes POSIX attributes between Active Directory and Directory Server entries.

NOTE

All POSIX attributes (such as ***uidNumber***, ***gidNumber***, and ***homeDirectory***) are synchronized between Active Directory and Directory Server entries. However, if a new POSIX entry or POSIX attributes are added to an existing entry in the Directory Server, *only the POSIX attributes are synchronized over to the Active Directory corresponding entry*. The POSIX object class (**posixAccount** for users and **posixGroup** for groups) is not added to the Active Directory entry.

12.7.1. Enabling POSIX Attribute Sync

The Posix Winsync API Plug-in is disabled by default and must be enabled for POSIX attributes to be synchronized from Active Directory user and group entries to the corresponding Directory Server entries.

1. Set the ***nsslapd-pluginEnabled*** attribute to **on**.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Posix Winsync API,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

NOTE

The precedence must be below 50 so that the Posix sync plug-in is loaded first. In the default configuration, the precedence is 25, and this value can remain the same in most deployments.

2. Restart the Directory Server to load the new configuration.

```
service dirsrv restart instance_name
```

12.7.2. Changing Posix Group Attribute Sync Settings

There are multiple plug-in attributes that can be set to control how the POSIX group attributes and group members are synced from the Active Directory entry to the corresponding Directory Server group and user entries. For details, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

The defaults can be used for most deployments, but the settings can be changed depending on the Active Directory environment. For example, to enable nested group mappings:

1. Use **ldapmodify** to change the attribute to the appropriate setting:

```
# ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Posix Winsync API,cn=plugins,cn=config
changetype: modify
replace: posixWinsyncMapNestedGrouping
posixWinsyncMapNestedGrouping: true
```

2. Restart the Directory Server to load the new configuration.

12.7.3. Changing to Older Versions of Windows Posix Attributes

By default, the Posix Winsync API Plug-in uses Posix schema for modern Active Directory servers: 2005, 2008, and later versions. There are slight differences between the modern Active Directory Posix schema and the Posix schema used by Windows Server 2003 and older Windows servers. If an Active Directory domain is using the older-style schema, then the Posix Winsync API Plug-in can be configured to use the older Microsoft System Services for Unix 3.0 (msSFU30) schema.

To switch to Windows 2003-style Posix schema:

1. Use **ldapmodify** to change the ***posixWinsyncMsSFUSchema*** attribute to **true**.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Posix Winsync API,cn=plugins,cn=config
changetype: modify
replace: posixWinsyncMsSFUSchema
posixWinsyncMsSFUSchema: true
```

2. Restart the Directory Server to load the new configuration.

```
service dirsrv restart instance_name
```

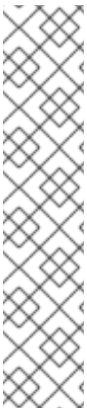
12.8. DELETING AND RESURRECTING ENTRIES

This section describes how enabling synchronization affects deleted entries on the sync peers and how resurrected entries are handled.

12.8.1. Deleting Entries

All changes on an Active Directory peers are always synced back to the Directory Server. This means that when an Active Directory group or user account is deleted on the Active Directory domain, the deletion is automatically synced back to the Directory Server sync peer server.

On Directory Server, on the other hand, when a Directory Server account is deleted, the corresponding entry on Active Directory is only deleted if the Directory Server entry has the ***ntUserDeleteAccount*** or ***ntGroupDeleteGroup*** attribute set to **true**.



NOTE

When a Directory Server entry is synchronized over to Active Directory for the first time, Active Directory automatically assigns it a unique ID. At the next synchronization interval, the unique ID is synchronized back to the Directory Server entry and stored as the ***ntUniqueId*** attribute. If the Directory Server entry is deleted on Active Directory *before* the unique ID is synchronized back to Directory Server, the entry *will not* be deleted on Directory Server. Directory Server uses the ***ntUniqueId*** attribute to identify and synchronize changes made on Active Directory to the corresponding Directory Server entry; without that attribute, Directory Server will not recognize the deletion.

To delete the entry on Active Directory and then synchronize the deletion over to Directory Server, wait the length of the ***winSyncInterval*** (by default, five minutes) after the entry is created before deleting it so that the ***ntUniqueId*** attribute is synchronized.

12.8.2. Resurrecting Entries

It is possible to add deleted entries back in Directory Server; the deleted entries are called *tombstone* entries. When a deleted entry which was synced between Directory Server and Active Directory is re-added to Directory Server, the resurrected Directory Server entry has all of its original attributes and values. This is called *tombstone reanimation*. The resurrected entry includes the original ***ntUniqueId*** attribute which was used to synchronize the entries, which signals to the Active Directory server that this new entry is a tombstone entry.

Active Directory resurrects the old entry and preserves the original unique ID for the entry.

For Active Directory entries, when the tombstone entry is resurrected on Directory Server, all of the attributes of the original Directory Server are retained and are still included in the resurrected Active Directory entry.

12.9. SENDING SYNCHRONIZATION UPDATES

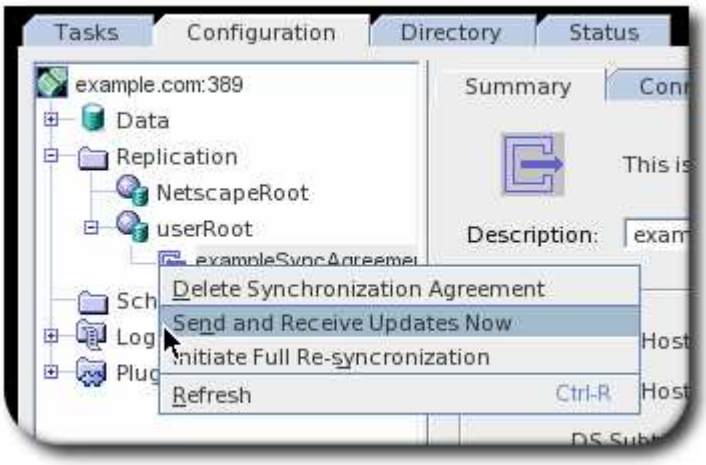
Synchronization occurs as frequently as is set in the ***winSyncInterval*** setting (for retrieving changes from the Active Directory domain) or ***nsds5replicaupdateschedule*** setting (for pushing changes from the Directory Server). By default, changes are retrieved from Active Directory every five minutes, and changes from the Directory Server are sent immediately.

A sync update can be triggered manually. It is also possible to do a full resynchronization, which sends and pulls every entry in the Directory Server and Active Directory as if it were new. A full resynchronization includes existing Directory Server entries which may not have previously been synchronized.

12.9.1. Performing a Manual Sync Update

During normal operations, all the updates made to entries in the Directory Server that need to be sent to Active Directory are collected the changelog and then replayed during an incremental update.

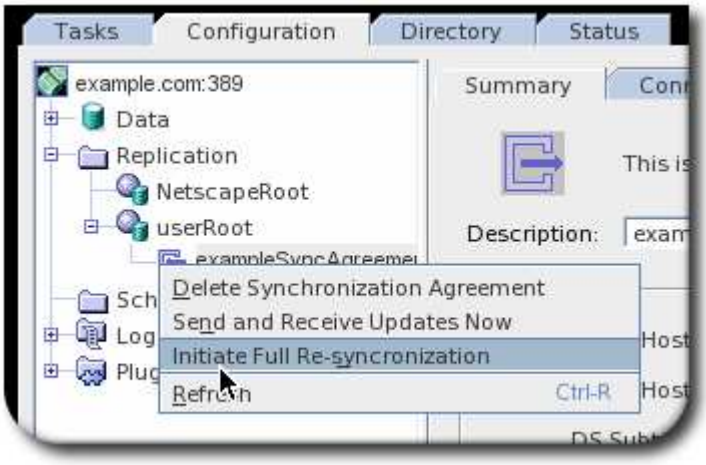
1. Go to the **Configuration** tab in the Console.
2. Open the **Replication** folder and expand the appropriate database.
3. Select the sync agreement.
4. Right-click on the agreement or open the **Object** menu.
5. Select **Send and Receive Updates** from the drop down menu.



12.9.2. Sending a Total Update (Full Synchronization)

If there have been major changes to data, or synchronization attributes are added to pre-existing Directory Server entries, it is necessary to initiate a *resynchronization*. Resynchronization is a total update; the entire contents of synchronized subtrees are examined and, if necessary, updated. Resynchronization is done without using the changelog. This is similar to initializing or reinitializing a consumer in replication.

1. Go to the **Configuration** tab in the Console.
2. Open the **Replication** folder and expand the appropriate database.
3. Select the sync agreement.
4. Right-click on the agreement or open the **Object** menu.
5. Select **Initialize Full Re-synchronization** from the drop down menu.



Resynchronizing will not delete data on the sync peer; it sends and receives all updates and add any new or modified Directory Server entries; for example, it adds a pre-existing Directory Server user that had the **ntUser** object class added.

12.9.3. Sending Sync Updates in the Command Line

To send sync updates through the command line, add the ***nsDS5BeginReplicaRefresh*** attribute to the sync agreement. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
add: nsDS5BeginReplicaRefresh
nsDS5BeginReplicaRefresh: start
```

This attribute is removed from the entry as soon as the update is complete.

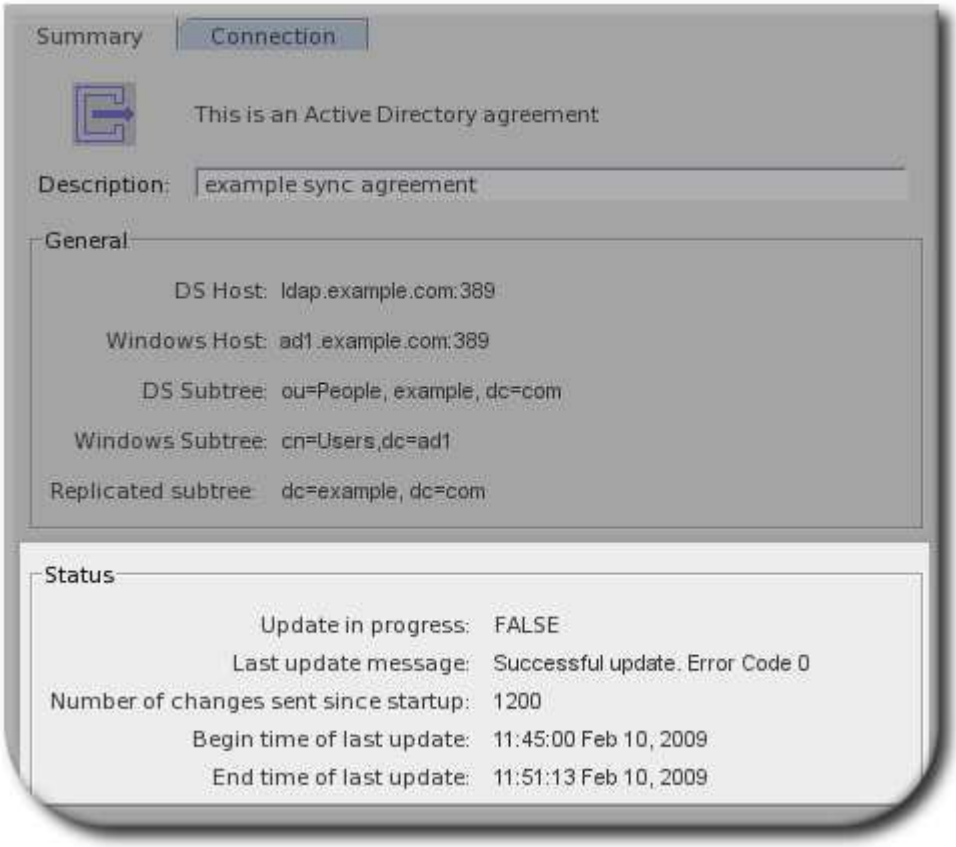
NOTE

This initiates a full synchronization for the entire database, not an incremental update of recent changes.

12.9.4. Checking Synchronization Status

Check synchronization status in the **Replication** tab in the **Status** of the Console. Highlight the synchronization agreement to monitor, and the relevant information should appear in the right-hand pane. The **Status** area shows whether the last incremental and total updates were successful and when they occurred.

1. Go to the **Configuration** tab in the Console.
2. Open the **Replication** folder and expand the appropriate database.
3. Select the sync agreement.
4. In the **Summary** tab, the status of the latest sync process is shown at the bottom.



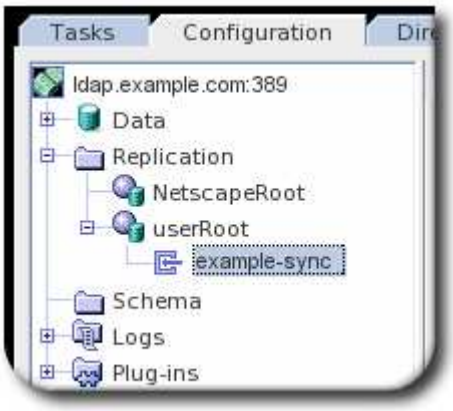
12.10. MODIFYING THE SYNC AGREEMENT

Certain attributes of the sync agreement can be modified, including the connection information. Using the command line, many additional parameters can be created with or added to the sync agreement, including changing the sync interval and setting a sync schedule.

12.10.1. Editing the Sync Agreement in the Console

Most of the information which can be edited in the Console is limited to connection information, including the protocol to use and the bind credentials. It is also possible to edit the sync agreement description.

1. In the **Configuration** tab, expand the **Replication** folder.
2. Expand the database being synced. All of the synchronization agreements are listed below the database. Double-click the sync agreement to open it in the main window.



3. Click the **Connection** tab.

There are three areas of information that can be edited.

- The connection type (standard, TLS/SSL, and Start TLS).
- The bind user, both DN and password.
- Whether to sync new Directory Server users and new Directory Server groups automatically.

There are three options for the connection type – standard, TLS/SSL, and Start TLS – but there are really only two connection protocols, LDAP and LDAPS. Both a standard connection and Start TLS connection use LDAP (Start TLS creates a secure connection over an insecure port).

It is *not* possible to change the connection protocol because it is not possible to change the port number used to connect to the Windows sync peer.

It is possible to change the connection type between the standard connection and Start TLS, but it is not possible to change from TLS/SSL to either the standard or Start TLS connections. Likewise, it is not possible to go from standard or Start TLS to TLS/SSL. If you need to change the connection protocol or the port number, delete the sync agreement and create a new one.

12.10.2. Adding and Editing the Sync Agreement in the Command Line

Creating or editing the sync agreement through the command line is more flexible and provides more options than using the Directory Server Console. The full list of sync agreement attributes are listed in Section 12.10.2.5, "Sync Agreement Attributes" and described in the *Configuration and Command-Line Tool Reference* .

12.10.2.1. Creating a Basic Sync Agreement

The most basic sync agreement defines the Directory Server database and the Active Directory sync peer:

- For the Directory Server database:

    - The synchronized subtree in the directory (***nsds7DirectoryReplicaSubtree***)
    - The Directory Server root DN (***nsDS5ReplicaRoot***)
    - The synchronized subtree in the directory (***nsds7DirectoryReplicaSubtree***)
- For the Active Directory domain:
    - The synchronized subtree in the Active Directory domain (***nsds7WindowsReplicaSubtree***)
    - The Active Directory domain name (***nsds7WindowsDomain***)

It also defines the connection information that the Directory Server uses to bind to the Active Directory domain:

- The Active Directory host name, IPv4 address, or IPv6 address (***nsDS5ReplicaHost***).
- The Active Directory port (***nsDS5ReplicaPort***).
- The type of connection (***nsDS5ReplicaTransportInfo***), which can be standard (**LDAP**), SSL (**SSL**), or Start TLS (**TLS**), which is a secure connection over a standard port.
- The user name (***nsDS5ReplicaBindDN***) and password (***nsDS5ReplicaCredentials***) for the Directory Server to use to bind to the Active Directory server.

For example:

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsDSWindowsReplicationAgreement
cn: ExampleSyncAgreement
nsds7WindowsReplicaSubtree: cn=Users,dc=ad1
nsds7DirectoryReplicaSubtree: ou=People,dc=example,dc=com
nsds7WindowsDomain: ad1
nsDS5ReplicaRoot: dc=example,dc=com
nsDS5ReplicaHost: ad1.windows-server.com
nsDS5ReplicaPort: 389
nsDS5ReplicaBindDN: cn=sync user,cn=config
nsDS5ReplicaCredentials: {DES}ffGad646dT0nnsT8nJOaMA==
nsDS5ReplicaTransportInfo: TLS
nsds7NewWinUserSyncEnabled: on
nsds7NewWinGroupSyncEnabled: on
```

12.10.2.2. Setting Sync Schedules

Synchronization works two ways. The Directory Server sends its updates to Active Directory on a configurable schedule, similar to replication, using the ***nsds5replicaupdateschedule*** attribute. The Directory Server polls the Active Directory to check for changes; the frequency that it checks the Active Directory server is set in the ***winSyncInterval*** attribute.

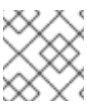
By default, the Directory Server update schedule is to always be in sync. The Active Directory interval is to poll the Active Directory every five minutes.

To change the schedule the Directory Server uses to send its updates to the Active Directory, edit the ***nsds5replicaupdateschedule*** attribute. The schedule is set with start (*SSSS*) and end (*EEEE*) times in the form *HHMM*, using a 24-hour clock. The days to schedule sync updates are use ranging from **0** (Sunday) to **6** (Saturday).

```
nsds5replicaupdateschedule: SSSS EEEE DDDDDDD
```

For example, this schedules synchronization to run from noon to 2:00pm on Sunday, Tuesday, Thursday, and Saturday:

```
nsds5replicaupdateschedule: 1200 1400 0246
```



NOTE

The synchronization times cannot wrap around midnight, so the setting **2300 0100** is not valid.

To change how frequently the Directory Server checks the Active Directory for changes to Active Directory entries, reset the ***winSyncInterval*** attribute. This attribute is set in seconds, so the default of **300** means that the Directory Server polls the Active Directory server every 300 seconds, or five minutes. Setting this to a higher value can be useful if the directory searches are taking too long and affecting performance.

```
winSyncInterval: 1000
```

12.10.2.3. Changing Sync Connections

Two aspects of the connection for the sync agreement can be altered:

- The bind user name and password (***nsDS5ReplicaBindDN*** and ***nsDS5ReplicaCredentials***).
- The connection method (***nsDS5ReplicaTransportInfo***).

  It is only possible to change the ***nsDS5ReplicaTransportInfo*** from **LDAP** to **TLS** and vice versa. It is not possible to change to or from **SSL** because it is not possible to change the port number, and switching between LDAP and LDAPS requires changing the port number.

For example:

```
nsDS5ReplicaBindDN: cn=sync user,cn=config
nsDS5ReplicaCredentials: {DES}ffGad646dT0nnsT8nJOaMA==
nsDS5ReplicaTransportInfo: TLS
```

WARNING

It is not possible to change the port number of the Active Directory sync peer. Therefore, it is also not possible to switch between standard/Start TLS connections and SSL connections, since that requires changing between standard and insecure ports.

To change to or from SSL/TLS, delete the sync agreement and add it again with the updated port number and new transport information.

12.10.2.4. Handling Entries That Move Out of the Synced Subtree

The sync agreement defines what subtrees in both Active Directory and Directory Server are synchronized between each other. Entries *within* the scope (the subtree) are synchronized; other entries are ignored.

However, the synchronization process actually starts at the root DN to begin evaluating entries for synchronization. Entries are correlated based on the ***samAccount*** in the Active Directory and the ***uid*** attribute in Directory Server. The synchronization plug-in notes if an entry (based on the ***samAccount***/***uid*** relationship) is removed from the synced subtree either because it is deleted or moved. That is the signal to the synchronization plug-in that the entry is no longer to be synced.

The issue is that the sync process needs some configuration to determine how to handle that moved entry. There are three options: delete the corresponding entry, ignore the entry (the default), or unsync the entry.

NOTE

These sync actions only relate to how to handle *on the Directory Server side* when an entry is moved out of scope on the Active Directory side. This does not affect any Active Directory entry if an entry is moved out of the synced subtree on the Directory Server side.

The default behavior in Directory Server 9.0 was to delete the corresponding Directory Server entry*This was true even if the entry on the Active Directory side was never synchronized over to the Directory Server side.* Starting in Directory Server 9.1, the default behavior is to ignore the entry and take no action.

For example, a user with the ***samAccount*** ID of **jsmith** was created in the **ou=Employees** subtree on Active Directory. The synchronized subtree is **ou=Users**, so the **jsmith** user was never synchronized over to Directory Server.

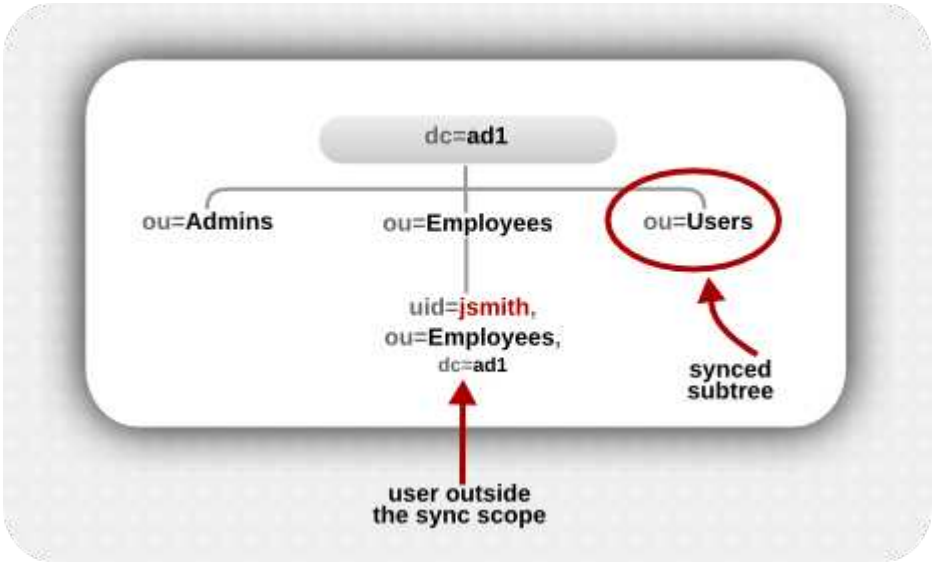


Figure 12.4. Active Directory Tree

For 7.x and 8.x versions of Directory Server, synchronization simply ignored that user, since it was outside the synced subtree.

Starting in Directory Server 9.0, Directory Server began supporting subtree renames – which means that existing entries could be moved between branches of the directory tree. The synchronization plug-in, then, assumes that entries in the Active Directory tree which correspond to a Directory Server user (***samAccount*/*uid*** relationship) but are outside the synced subtree are *intentionally* moved outside the synced subtree – essentially, a rename operation. The assumption then was that the "corresponding" Directory Server entry should be deleted.

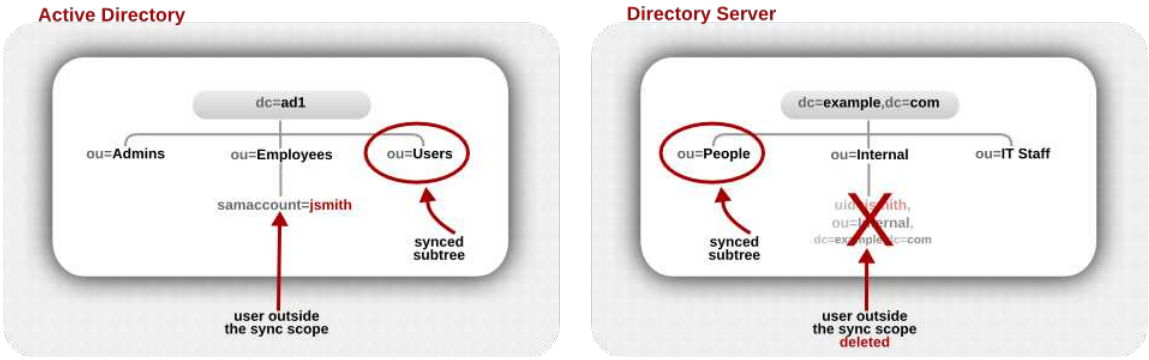


Figure 12.5. Active Directory and Directory Server Trees Compared

This assumption is not necessarily an accurate one, particularly for user entries which always existed outside the synced subtree.

The ***winSyncMoveAction*** attribute for the synchronization agreement sets instructions on how to handle these moved entries:

- **none** takes no action, so if a synced Directory Server entry exists, it may be synced over to or create an Active Directory entry *within* scope. If no synced Directory Server entry exists, nothing happens at all (this is the default behavior in Directory Server 9.1).
- **unsync** removes any sync-related attributes (***ntUser*** or ***ntGroup***) from the Directory Server entry but otherwise leaves the Directory Server entry intact.

  IMPORTANT

  There is a risk when unsyncing entries that the Active Directory entry may be deleted at a later time, and the Directory Server entry will be left intact. This can create data inconsistency issues, especially if the Directory Server entry is ever used to recreate the entry on the Active Directory side later.

- **delete** deletes the corresponding entry on the Directory Server side, regardless of whether it was ever synced with Active Directory (this was the default behavior in 9.0).

IMPORTANT

You almost never want to delete a Directory Server entry without deleting the corresponding Active Directory entry. This option is available only for compatibility with Directory Server 9.0 systems.

If it is necessary to change the default behavior from **none**, then edit the synchronization agreement to add the ***winSyncMoveAction*** attribute:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=ExampleSyncAgreement,cn=sync replica,cn=dc=example\,dc=com,cn=mapping tree,cn=config
changetype: modify
add: winSyncMoveAction
winSyncMoveAction: unsync
```

12.10.2.5. Sync Agreement Attributes

The common sync agreement attributes are listed in Table 12.6, "Sync Agreement Attributes". All of the possible sync agreement attributes are described in detail in the *Red Hat Directory Server 9 Configuration, Command, and File Reference*.

Table 12.6. Sync Agreement Attributes

Object Class or Attribute	Description
nsDSWindowsReplicationAgreement	An operational object class which contains the sync agreement attributes.
cn	Gives the name for the sync agreement.
nsds7WindowsReplicaSubtree	Specifies the Windows server suffix (root or sub) that is synced.
nsds7DirectoryReplicaSubtree	Specifies the Directory Server suffix (root or sub) that is synced.
nsds7NewWinUserSyncEnabled	Sets whether new Windows user accounts are automatically created on the Directory Server.
nsds7NewWinGroupSyncEnabled	Specifies whether new Windows group accounts are automatically created on the Directory Server.
nsds7WindowsDomain	Identifies the Windows domain being synchronized; analogous to ***nsDS5ReplicaHost*** in a replication agreement.
nsds5replicaport	Gives the LDAP port for the Windows server. To use TLS/SSL, give the secure port number (**636** by default) and set the ***nsds5ReplicaTransportInfo*** attribute to **SSL**. To use Start TLS, which initiates a secure connection over a standard port, use the standard port, **389**, with the ***nsds5ReplicaTransportInfo*** attribute to **TLS**.
nsds5replicatransportinfo	To use TLS/SSL, set this parameter to **SSL**. To use Start TLS, which initiates a secure connection over a standard port, set this parameter to **TLS**. To use simple authentication, set this parameter to **LDAP**.
nsds5ReplicaBindDN	The sync user DN used by the Directory Server instance to bind to the Windows server.

Object Class or Attribute	Description
nsds5replicabindmethod	The connection type for replication between the servers. The connection type defines how the supplier authenticates to the consumer. Leaving the bind method empty or setting it to **SIMPLE** means that the server uses basic password-based authentication. This requires the ***nsds5ReplicaBindDN*** and ***nsds5ReplicaCredentials*** attributes to give the bind information. The **SSLCLIENTAUTH** option uses a secure connection. This requires setting the ***nsds5ReplicaTransportInfo*** attribute be set to **SSL** or **TLS**.
nsDS5ReplicaCredentials	*Only for simple authentication.* Stores the hashed password used with the bind DN given for simple authentication.
nsds5replicaroot	Sets which Directory Server subtree is synchronized. Usually, it is recommended that the synchronized subtree be high in the directory tree so that the entire database is synchronized. For example: `dc=example,dc=com`
description	A text description of the replication agreement. Make this a useful description so it is easier to manage sync agreements.
nsds5replicaupdateschedule	Sets the start and end time for the replication updates and the days on which replication occurs in the form *start_time end_time days*. If the schedule is omitted, synchronization occurs all of the time.
oneWaySync	*Optional.* Configures synchronization updates to come from only one direction, either from Active Directory to Directory Server (**fromWindows**) or from Directory Server to Active Directory (**toWindows**). This attribute is not set by default, so synchronization is bi-directional.
winSyncInterval	*Optional.* Sets how frequently, in seconds, the Directory Server polls the Windows server for updates to write over. If this is not set, the default is **300**, which is 300 seconds or five (5) minutes.
winSyncMoveAction	*Optional.* Sets how the sync plug-in handles corresponding entries that are discovered in Active Directory outside of the synced subtree. The sync process can ignore these entries (none, the default) or it can assume that the entries were moved intentionally to remove them from synchronization, and it can then either delete the corresponding Directory Server entry (delete) or remove the synchronization attributes and no longer sync the entry (unsync).
nsds5BeginReplicaRefresh	*Optional.* Performs an online (immediate) initialization of the sync peer. If this is set, the attribute is only present as long as the sync peer is being updated initialized; when the

12.11. MANAGING THE PASSWORD SYNC SERVICE

IMPORTANT

Password Sync must be installed on every domain controller in the Active Directory domain in order to synchronize Windows passwords.

The service synchronizes password changes made on Active Directory with the corresponding entries' passwords on the Directory Server. Like any Windows service, it can be modified, started and stopped, and uninstalled, depending on how synchronization between Directory Server and Active Directory changes.

The Password Sync Service is supported on Microsoft Windows Server 2008 R2 (32-bit and 64-bit).

12.11.1. Modifying Password Sync

To reconfigure Password Sync:

1. Open **Control Panel**, and double-click **Add/Remove Programs**.
2. Click the **Change** button to relaunch the installer to change the settings.

3. Go back through the configuration screens to make any changes to the configuration.

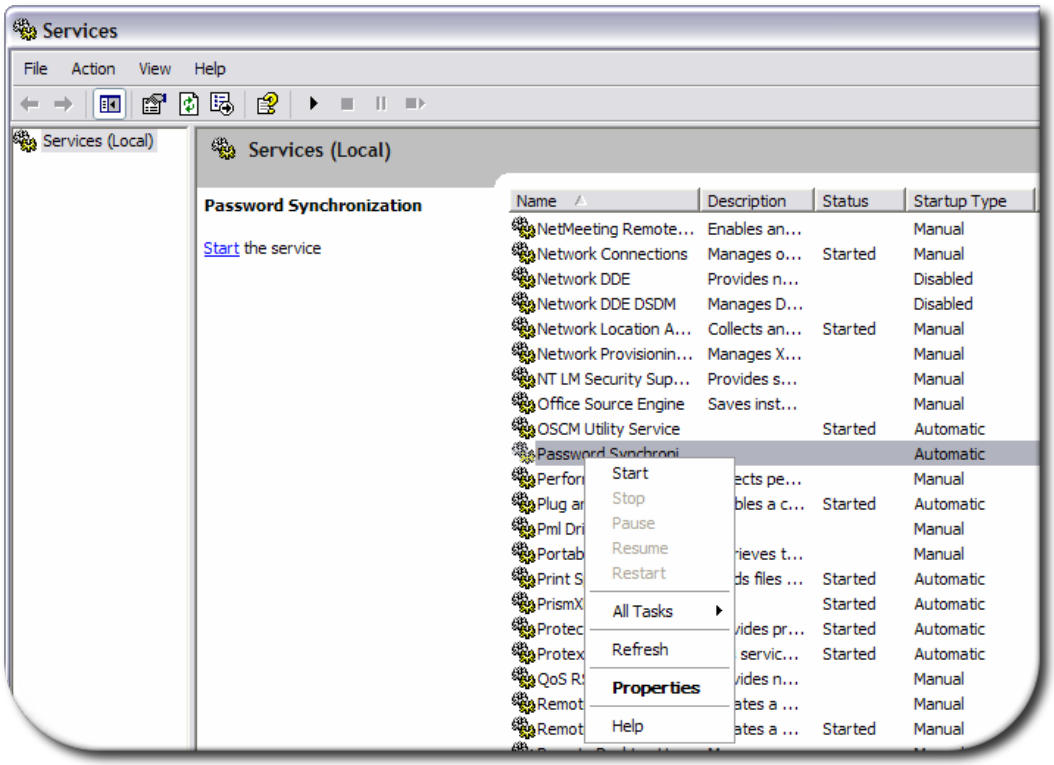
12.11.2. Starting and Stopping the Password Sync Service

The Password Sync Service is configured to start whenever the Active Directory host is started. To reconfigure the service so that it does not start when Windows reboots:

1. Go to the **Control Panel**, and select **Services**.
2. Scroll through the list of services for the Password Sync Service. The **Startup** field is set to **Automatic**.
3. Double-click Password Sync.
4. Select the **Manual** radio button, and then click **OK**.

To start and stop Password Sync:

1. Go to the **Control Panel**, and select **Services**.
2. Scroll through the list of services for Password Sync, and right-click.
3. Select **Stop**, **Start**, or **Restart**, and hit okay.



It's also possible to select the sync service and then click the start or stop links in the upper left of the **Services** window.

Changed passwords are captured even if Password Sync is not running. If Password Sync is restarted, the password changes are sent to Directory Server at the next synchronization.

12.11.3. Uninstalling Password Sync Service

1. Open **Control Panel**, and double-click **Add/Remove Programs**.
2. Select click **Remove** to uninstall the Password Sync Service.

3. If SSL was configured for the Password Sync, then the **cert8.db** and **key3.db** databases that were created were not removed when Password Sync was uninstalled. Delete these files by hand.

12.11.4. Upgrading Password Sync

For details, see the corresponding section in the *Red Hat Directory Server Installation Guide*.

12.12. TROUBLESHOOTING

If synchronization does not seem to function properly, see the Windows event log or Directory Server error log for information on any potential problems.

Enable replication logging to record synchronization errors

Enable replication logging for more detailed information on synchronization to be recorded in the error logs. The replication log level produces more verbose logs from the sync code. Messages related to synchronization traffic (which is the same as replication traffic) can help in diagnosing problems.

1. In the Console, click the **Configuration** tab.
2. Select **Logs** from the navigation menu on the right, and open the error log.
3. Scroll down to error log level, and select **Replication** from the menu.
4. Hit save.

Error #1: The message box when creating the sync agreement indicates that the it cannot connect to Active Directory.

Make sure that the directory suffixes, Windows domain and domain host, and the administrator DN and password are correct. Also verify that the port number used for LDAPS is correct. If all of the connection information is correct, make sure that Active Directory machine is running.

Error #2: After synchronization, the status returns error 81.

One of the sync peer servers has not been properly configured for SSL communication. Examine the Directory Server access log file to see if the connection attempt was received by the Directory Server. There are also helpful messages in the Directory Server's error log file.

To narrow down the source of the misconfiguration, try to establish an LDAPS connection to the Directory Server. If this connection attempt fails, check all values (including the port number, host name or IPv4/IPv6 address, search base, and user credentials) to see if any of these are the problem. If all else fails, reconfigure the Directory Server

with a new certificate.

If the LDAPS connection to the Directory Server is successful, it is likely that the misconfiguration is on Active Directory. Examine the Windows event log file for error messages.

NOTE

A common problem is that the certificate authority was not configured as trusted when the Windows sync services certificate database was configured.

Error #3: An entry is moved from one subtree on Active Directory to another subtree, but the user is not moved to the corresponding subtree on Directory Server.

This is a known issue with synchronizing modrdn operations on Active Directory with entries on Directory Server. To work around it, delete the entry on Active Directory and then add it anew to the new subtree. The deletion and the addition will be properly synchronized to the Directory Server peer.

---

[3] Unlike a consumer, changes can still be made on the un-synced server. Use ACLs to prevent editing or deleting entries on the un-synced server to maintain data integrity.

CHAPTER 13. MANAGING ACCESS CONTROL

Red Hat Directory Server allows you to control access to your directory. This chapter describes the how to implement *access control*. To take full advantage of the power and flexibility of access control, while you are in the planning phase for your directory deployment, define an access control strategy as an integral part of your overall security policy.

13.1. ACCESS CONTROL PRINCIPLES

The mechanism which defines user access is called *access control*. When the server receives a request, it uses the authentication information provided by the user in the bind operation and the access control instructions (ACIs) defined in the server to allow or deny access to directory information. The server can allow or deny permissions for actions on entries like read, write, search, and compare. The permission level granted to a user may depend on the authentication information provided.

Access control in Directory Server is flexible enough to provide very precise rules on when the ACIs are applicable:

- For the entire directory, a subtree of the directory, specific entries in the directory (including entries defining configuration tasks), or a specific set of entry attributes.
- For a specific user, all users belonging to a specific group or role, or all users of the directory.
- For a specific location such as an IP address or a DNS name.

13.1.1. ACI Structure

Access control instructions are stored in the directory as attributes of entries. The **aci** attribute is an operational attribute; it is available for use on every entry in the directory, regardless of whether it is defined for the object class of the entry. It is used by the Directory Server to evaluate what rights are granted or denied when it receives an LDAP request from a client. The ***aci*** attribute is returned in an **ldapsearch** operation if specifically requested.

The three main parts of an ACI statement are:

- Target
- Permission
- Bind Rule

The permission and bind rule portions of the ACI are set as a pair, also called an *access control rule* (ACR). The specified permission is granted or denied depending on whether the accompanying rule is evaluated to be true.

13.1.2. ACI Placement

If an entry containing an ACI does not have any child entries, the ACI applies to that entry only. If the entry has child entries, the ACI applies to the entry itself and all entries below it. As a direct consequence, when the server evaluates access permissions to any given entry, it verifies the ACIs for every entry between the one requested and the directory suffix, as well as the ACIs on the entry itself.

The ***aci*** attribute is multi-valued, which means that you can define several ACIs for the same entry or subtree.

An ACI created on an entry can be set so it does not apply directly to that entry but to some or all of the entries in the subtree below it. The advantage of this is that general ACIs can be placed at a high level in the directory tree that effectively apply to entries more likely to be located lower in the tree. For example, an ACI that targets entries that include the **inetorgperson** object class can be created at the level of an **organizationalUnit** entry or a **locality** entry.

Minimize the number of ACIs in the directory tree by placing general rules at high level branch points. To limit the scope of more specific rules, place them as close as possible to leaf entries.

NOTE

ACIs placed in the root DSE entry apply only to that entry.

13.1.3. ACI Evaluation

To evaluate the access rights to a particular entry, the server compiles a list of the ACIs present on the entry itself and on the parent entries back up to the top level entry stored on the Directory Server. ACIs are evaluated across all of the databases for a particular Directory Server but not across all Directory Server instances.

The evaluation of this list of ACIs is done based on the semantics of the ACIs, not on their placement in the directory tree. This means that ACIs that are close to the root of the directory tree do not take precedence over ACIs that are closer to the leaves of the directory tree.

For Directory Server ACIs, the *precedence rule* is that ACIs that deny access take precedence over ACIs that allow access. Between ACIs that allow access, union semantics apply, so there is no precedence.

For example, if you deny write permission at the directory's root level, then none of the users can write to the directory, regardless of the specific permissions you grant them. To grant a specific user write permissions to the directory, you have to restrict the scope of the original denial for write permission so that it does not include the user.

13.1.4. ACI Limitations

When creating an access control policy for your directory service, you need to be aware of the following restrictions:

- If your directory tree is distributed over several servers using the chaining feature, some restrictions apply to the keywords you can use in access control statements:
  - ACIs that depend on group entries (**groupdn** keyword) must be located on the same server as the group entry. If the group is dynamic, then all members of the group must have an entry on the server, too. If the group is static, the members' entries can be located on remote servers.
  - ACIs that depend on role definitions (**roledn** keyword) must be located on the same server as the role definition entry. Every entry that is intended to have the role must also be located on the same server.

  However, you can match values stored in the target entry with values stored in the entry of the bind user; for example, using the **userattr** keyword. Access is evaluated normally even if the bind user does not have an entry on the server that holds the ACI.

  For more information on how to chain access control evaluation, see Section 2.3.6, "Database Links and Access Control Evaluation".
- Attributes generated by class of service (CoS) cannot be used in all ACI keywords. Specifically, you should not use attributes generated by CoS with the following keywords:
  - targetfilter (Section 13.3.2.4, "Targeting Entries or Attributes Using LDAP Filters")
  - targetattrfilters (Section 13.3.2.2, "Targeting Attributes")
  - userattr (Section 13.4.5.1, "Using the userattr Keyword")

  If you create target filters or bind rules that depend on the value of attributes generated by CoS, the access control rule will not work. For more information on CoS, see Chapter 6, *Organizing and Grouping Entries*.
- Access control rules are always evaluated on the local server. Therefore, it is not necessary to specify the host name or port number of the server in LDAP URLs used in ACI keywords. If you do, the LDAP URL is not taken into account at all. For more information on LDAP URLs, see Appendix C, *LDAP URLs*.

13.2. DEFAULT ACIS

When the Admin Server is set up, the following default ACIs apply to the directory information stored in the **userRoot** database:

- Users can modify a list of common attributes in their own entries, including the ***mail***, ***telephoneNumber***, ***userPassword***, and ***seeAlso*** attributes. Operational and most of the security attributes, such as ***aci***, ***nsroledn***, and ***passwordExpirationTime***, cannot be modified by users.
- Users have anonymous access to the directory for search, compare, and read operations.
- The administrator (by default **uid=admin,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot**) has all rights except proxy rights.
- All members of the **Configuration Administrators** group have all rights except proxy rights.
- All members of the **Directory Administrators** group have all rights except proxy rights.
- **Server Instance Entry** (SIE) group.

The **NetscapeRoot** subtree has its own set of default ACIs:

- All members of the **Configuration Administrators** group have all rights on the **NetscapeRoot** subtree except proxy rights.
- Users have anonymous access to the **NetscapeRoot** subtree for search and read operations.
- All authenticated users have search, compare, and read rights to configuration attributes that identify the Admin Server.
- Group expansion.

The following sections explain how to modify these default settings.

13.3. CREATING ACIS MANUALLY

You can create access control instructions manually using LDIF statements and add them to the directory tree using the **ldapmodify** utility, similar to the instructions in Section 3.3, "Using LDIF Update Statements to Create or Modify Entries".

NOTE

LDIF ACI statements can be very complex. However, if you are setting access control for a large number of directory entries, using LDIF is the preferred because it is faster than using the Console. To familiarize yourself with LDIF ACI statements, however, you may want to use the Directory Server Console to set the ACI and then click the **Edit Manually** button on the **Access Control Editor**. This shows you the correct LDIF syntax. If your operating system allows it, you can even copy the LDIF from the **Access Control Editor** and paste it into your LDIF file.

13.3.1. The ACI Syntax

The ***aci*** attribute uses the following syntax:

```
aci: (target)(version 3.0;acl "name";permissionbind_rules;)
```

- *target* specifies the entry, attributes, or set of entries and attributes for which to control access. The target can be a distinguished name, one or more attributes, or a single LDAP filter. The target is an optional part of the ACI.
- **version 3.0** is a required string that identifies the ACI version.
- *name* is a name for the ACI. The name can be any string that identifies the ACI. The ACI name is required.
- *permission* specifically outlines what rights are being allowed or denied; for example, read or search rights.
- *bind_rules* specify the credentials and bind parameters that a user has to provide to be granted access. Bind rules can also specifically deny access to certain users or groups of users.

You can have multiple permission-bind rule pairs for each target. This allows you to set multiple access controls for a given target efficiently. For example:

```
target(permissionbind_rule)(permissionbind_rule)...
```

If you have several ACRs in one ACI statement, the syntax is in the following form:

```
aci: (target)(version 3.0;acl "name";permissionbind_rule; permissionbind_rule; ... permissionbind_rule;)
```

The following is an example of a complete LDIF ACI:

```
aci: (target="ldap:///uid=bjensen,dc=example,dc=com")(targetattr=*)
(version 3.0;acl "aci1";allow (write) userdn="ldap:///self";)
```

In this example, the ACI states that the user **bjensen** has rights to modify all attributes in her own directory entry.

13.3.2. Defining Targets

The target identifies to what the ACI applies. If the target is not specified, the ACI applies to the entry containing the ***aci*** attribute and to the entries below it. A target can be any of the following:

- A directory entry or all of the entries in a subtree, as described in Section 13.3.2.1, "Targeting a Directory Entry".

- Attributes of an entry, as described in Section 13.3.2.2, "Targeting Attributes".
- A set of entries or attributes that match a specified LDAP filter, as described in Section 13.3.2.4, "Targeting Entries or Attributes Using LDAP Filters".
- An attribute value, or a combination of values, that match a specified LDAP filter, as described in Section 13.3.2.5, "Targeting Attribute Values Using LDAP Filters" .

The general syntax for a target is as follows:

```
(keyword = "expression")
(keyword != "expression")
```

- *keyword* indicates the type of target.
- equal (=) indicates that the target is the object specified in the *expression*, and not equal (!=) indicates the target is not the object specified in the *expression*.
- *expression* identifies the target.

The quotation marks (**""**) around *expression* are required. What you use for *expression* is dependent upon the *keyword* that you supply.

Table 13.1, "LDIF Target Keywords" lists each keyword and the associated expressions.

Table 13.1. LDIF Target Keywords

Keyword	Valid Expressions	Wildcard Allowed
target	ldap:///*distinguished_name*	Yes
targetattr	*attribute*	Yes
targetfilter	*LDAP_filter*	Yes
targetattrfilters	*operation=attribute:LDAP_filter*[a]	Yes
[a] Supported operations: **add** and **del**.		

In all cases, you must keep in mind that when you place an ACI on an entry, if it is not a leaf entry, the ACI also applies to all entries below it. For example, if you target the entry **ou=accounting,dc=example,dc=com**, the permissions you set apply to all entries in the accounting branch of the **example** tree.

As a counter example, if you place an ACI on the **ou=accounting,dc=example,dc=com** entry, you cannot target the **uid=sarette,ou=people,dc=example,dc=com** entry because it is not located under the accounting tree.

Be wary of using **!=** when specifying an attribute to deny. ACLs are treated as a logical OR, which means that if you created two ACLs as shown below, the result allows all values of the target attribute.

```
acl1: ( target=...)( targetattr!=a )(version 3.0; acl "name";allow (...)..
acl2: ( target=...)( targetattr!=b )(version 3.0; acl "name";allow (...)..
```

The first ACL (**acl1**) allows **b** and the second ACL (**acl2**) allows **a**. The result of these two ACLs is the same as the one resulting from using an ACL of the following form:

```
acl3: ( targetattr="*" ) allow (...) ...
```

In the second example, nothing is denied, which could give rise to security problems.

When you want to deny access to a particular attribute, use **deny** in the permissions clause rather than using **allow** with **(targetattr != value)**. For example, usages such as these are recommended:

```
acl1: ( target=...)( targetattr=a )(version 3.0; acl "name";deny (...)..
acl2: ( target=...)( targetattr=b )(version 3.0; acl "name";deny (...)..
```

13.3.2.1. Targeting a Directory Entry

To target a directory entry (and the entries below it), you must use the **target** keyword. The **target** keyword can accept a value of the following format:

```
target="ldap:///distinguished_name
```

This identifies the distinguished name of the entry to which the access control rule applies. For example:

```
(target = "ldap:///uid=bjensen,dc=example,dc=com")
```

NOTE

If the DN of the entry to which the access control rule applies contains a comma, escape the comma with a single backslash (\), such as **(target="ldap:///uid=lfuentes,dc=example.com Bolivia\,S.A.")**.

Wildcards can be used when targeting a distinguished name using the **target** keyword. The wildcard indicates that any character or string or substring is a match for the wildcard. Pattern matching is based on any other strings that have been specified with the wildcard.

The following are legal examples of wildcard usage:

- **(target="ldap:///uid=*,dc=example,dc=com")** – Matches every entry in the entire **example** tree that has the ***uid*** attribute in the entry's RDN.
- **(target="ldap:///uid=*Anderson,dc=example,dc=com")** – Matches every entry directly under the **example** node with a **uid** ending in Anderson.
- **(target="ldap:///uid=C*A,dc=example,dc=com")** – Matches every entry directly under the **example** node with a ***uid*** beginning with C and ending with A.
- **(target="ldap:///uid=*,dc=example,dc=com")** – Matches every entry in the entire **example** tree that has the ***uid*** attribute in the entry's RDN.
- **(target="ldap:///uid=*,ou=*,dc=example,dc=com")** – Matches every entry in the **example** tree whose distinguished name contains the ***uid*** and ***ou*** attributes. Thus, **uid=fchen,ou=Engineering,dc=example,dc=com** or **uid=claire,ou=Engineering,ou=people,dc=example,dc=com** would match, but **uid=bjensen,dc=example,dc=com ou=Engineering,dc=example,dc=com** would not.

Depending on the position of the wildcard, it can apply to the full DN, not only to attribute values. Therefore, the wildcard can be used as a substitute for portions of the DN. For example, **uid=andy*,dc=example,dc=com** targets all the directory entries in the entire **example** tree with a matching uid attribute and not just the entries that are immediately below the **dc=example,dc=com** node. In other words, this target matches with longer expressions such as **uid=andy,ou=eng,dc=example,dc=com** or **uid=andy,ou=marketing,dc=example,dc=com**.

NOTE

You cannot use wildcards in the suffix part of a distinguished name. That is, if your directory uses the suffixes **c=US** and **c=GB**, then you cannot use **(target="ldap:///dc=example,c=*")** as a target to reference both suffixes. Neither can you use a target such as **uid=bjensen,dc=*.com**.

13.3.2.2. Targeting Attributes

In addition to targeting directory entries, you can also target one or more attributes included in the targeted entries. This is useful to deny or allow access to partial information about an entry. For example, you could allow access to only the common name, surname, and telephone number attributes of a given entry while denying access to sensitive information such as passwords.

You can specify that the target is equal (**=**) or is not equal (**!=**) to a specific attribute. The attributes you supply do not need to be defined in the schema. This absence of schema checking makes it possible to implement an access control policy when you set up your directory service for the first time, even if the ACLs you create do not apply to the current directory content.

To target attributes, use the **targetattr** keyword. The keyword uses the following syntax:

```
(targetattr = "attribute")
```

For example, to target the common name (***cn***) attribute:

```
(targetattr = "cn")
```

You can target multiple attributes by using the **targetattr** keyword with the following syntax:

```
(targetattr = "attribute1 || attribute2 || ...")
```

To target all attributes:

```
(targetattr = "*")
```

NOTE

If the **targetattr** keyword is not set, the rights granted by the ACI applies to no attributes.

The attributes specified in the **targetattr** keyword apply to the entry that the ACI is targeting and to all the entries below it. If you target the password attribute on the entry **uid=bjensen,ou=Marketing,dc=example,dc=com**, only the password attribute on the **bjensen** entry is affected by the ACI because it is a leaf entry.

If, however, you target the tree's branch point **ou=Marketing,dc=example,dc=com**, then all the entries beneath the branch point that can contain a password attribute are affected by the ACI.

13.3.2.3. Targeting Both an Entry and Attributes

By default, the entry targeted by an ACI containing a **targetattr** keyword is the entry on which the ACI is placed. That is, putting an ACI such as **aci: (targetattr = "uid")(***access_control_rules***;)** on the **ou=Marketing,dc=example,dc=com** entry means that the ACI applies to the entire **Marketing** subtree. However, you can also explicitly specify a target using the **target** keyword:

```
aci: (target="ldap:///ou=Marketing,dc=example,dc=com")(targetattr="uid")(access_control_rules;)
```

The order in which you specify the **target** and the **targetattr** keywords is not important.

13.3.2.4. Targeting Entries or Attributes Using LDAP Filters

You can use LDAP filters to target a group of entries that match certain criteria. To do this, you must use the **targetfilter** keyword with an LDAP filter. The syntax of the **targetfilter** keyword is as follows:

```
(targetfilter = "LDAP_filter")
```

LDAP_filter is a standard LDAP search filter. For more information on the syntax of LDAP search filters, see Chapter 10, *Finding Directory Entries*.

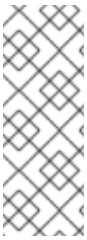
For example, suppose that all entries in the accounting department include the attribute-value pair **ou=accounting**, and all entries in the engineering department include the attribute-value pair **ou=engineering** subtree. The following filter targets all the entries in the accounting and engineering branches of the directory tree:

```
(targetfilter = "(|(ou=accounting)(ou=engineering))")
```

This type of filter targets whole entries. You can associate the **targetfilter** and the **targetattr** keywords to create ACIs that apply to a subset of attributes in the targeted entries.

The following LDIF example allows members of the Engineering Admins group to modify the ***departmentNumber*** and ***manager*** attributes of all entries in the **Engineering** business category. This example uses LDAP filtering to select all entries with ***businessCategory*** attributes set to **Engineering**:

```
dn: dc=example,dc=com
objectClass: top
objectClass: organization
aci: (targetattr="departmentNumber || manager")
  (targetfilter="(businessCategory=Engineering)")
  (version 3.0; acl "eng-admins-write"; allow (write)
  groupdn ="ldap:///cn=Engineering Admins,dc=example,dc=com";)
```



NOTE

Although using LDAP filters can be useful when you are targeting entries and attributes that are spread across the directory, the results are sometimes unpredictable because filters do not directly name the object for which you are managing access. The set of entries targeted by a filtered ACI is likely to change as attributes are added or deleted. Therefore, if you use LDAP filters in ACIs, you should verify that they target the correct entries and attributes by using the same filter in an **ldapsearch** operation.

13.3.2.5. Targeting Attribute Values Using LDAP Filters

You can use access control to target specific attribute values. This means that you can grant or deny permissions on an attribute if that attribute's value meets the criteria defined in the ACI. An ACI that grants or denies access based on an attribute's value is called a value-based ACI.

For example, you might grant all users in your organization permission to modify the ***nsroledn*** attribute in their own entry. However, you would also want to ensure that they do not give themselves certain key roles, such as **Top Level Administrator**. LDAP filters are used to check that the conditions on attribute values are satisfied.

To create a value-based ACI, use the **targetattrfilters** keyword with the following syntax:

- For one operation with one attribute and filter combination:

  (targetattrfilters="*operation=attribute:filter*")

- For one operation with multiple attribute and filter combinations:

  (targetattrfilters="*operation=attribute_1:filter_1* && *attribute_2:filter_2* ... && *attribute_m:filter_m*")

- For two operations, each with multiple attribute and filter combinations:

  (targetattrfilters="*operation_1=attribute_1_1:filter_1_1* && *attribute_1_2:filter_1_2* ... && *attribute_1_m:filter_1_m* , *operation_2=attribute_2_1:filter_2_1* && *attribute_2_2:filter_2_2* ... & *attribute_2_n:filter_2_n* ")

In the previous syntax examples, you can set the operations either to **add** or **del**. The **attribute:filter** combination sets the filter and the attribute the filter is applied to.

When creating an entry, if a filter applies to an attribute in the new entry, then each instance of that attribute must satisfy the filter. When deleting an entry, if a filter applies to an attribute in the entry, then each instance of that attribute must also satisfy the filter.

When modifying an entry, if the operation adds an attribute, then the add filter that applies to that attribute must be satisfied; if the operation deletes an attribute, then the delete filter that applies to that attribute must be satisfied. If individual values of an attribute already present in the entry are replaced, then both the add and delete filters must be satisfied.

For example, consider the following attribute filter:

```
(targetattrfilters="add=nsroledn:(!(nsroledn=cn=superAdmin)) && telephoneNumber:(telephoneNumber=123*)")
```

This filter can be used to allow users to add any role (***nsroledn*** attribute) to their own entry, except the **superAdmin** role. It also allows users to add a telephone number with a 123 prefix.

NOTE

You cannot create value-based ACIs from the Directory Server Console.

13.3.2.6. Targeting a Single Directory Entry

Targeting a single directory entry is not straightforward because it goes against the design philosophy of the access control mechanism. However, it can be done in either of two ways:

- By creating a bind rule that matches user input in the bind request with an attribute value stored in the targeted entry. For more details, see Section 13.4.5, “Defining Access Based on Value Matching” .
- By using the **targetattr** and **targetfilter** keywords.

You can use the **targetattr** keyword to specify an attribute that is only present in the entry you want to target, and

not in any of the entries below your target. For example, if you want to target **ou=people,dc=example,dc=com**, and there are not any organizational units (***ou***) defined below that node, you could specify an ACI that contains **targetattr=ou**.

A safer method is to use the **targetfilter** keyword and to specify explicitly an attribute value that appears in the entry alone. For example, during the installation of the Directory Server, the following ACI is created:

```
aci: (targetattr="*")(targetfilter=(o=NetscapeRoot))(version 3.0;
    acl "Default anonymous access"; allow (read, search) userdn="ldap:///anyone";)
```

This ACI can apply only to the **o=NetscapeRoot** entry.

The risk associated with these method is that your directory tree might change in the future, and you would have to remember to modify this ACI.

13.3.3. Defining Permissions

Permissions specify the type of access you are allowing or denying. You can either allow or deny permission to perform specific operations in the directory. The various operations that can be assigned are known as *rights*.

There are two parts to setting permissions:

- Allowing or denying access
- Assigning rights

13.3.3.1. Allowing or Denying Access

You can either explicitly allow or deny access permissions to the directory tree.

NOTE

From the Directory Server Console, you cannot explicitly deny access, only grant permissions.

13.3.3.2. Assigning Rights

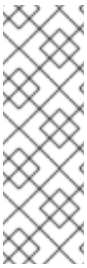
Rights detail the specific operations a user can perform on directory data. You can allow or deny all rights, or you can assign one or more of the following rights:

Table 13.2. User Rights

Right	Description
Read	Indicates whether users can read directory data. This permission applies only to the search operation.
Write	Indicates whether users can modify an entry by adding, modifying, or deleting *attributes*. This permission applies to the modify and modrdn operations.
Add	Indicates whether users can create an **entry**. This permission applies only to the add operation.
Delete	Indicates whether users can delete an **entry**. This permission applies only to the delete operation.
Search	Indicates whether users can search for the directory data. Users must have Search and Read rights in order to view the data returned as part of a search result. This permission applies only to the search operation.

Right	Description
Compare	Indicates whether the users can compare data they supply with data stored in the directory. With compare rights, the directory returns a success or failure message in response to an inquiry, but the user cannot see the value of the entry or attribute. This permission applies only to the compare operation.
Selfwrite	Indicates whether users can add or delete their own DN from a group. This right is used only for group management.
Proxy	Indicates whether the specified DN can access the target with the rights of another entry.
All	Indicates that the specified DN has all rights (**read**, **write**, **search**, **delete**, **compare**, and **selfwrite**) to the targeted entry, *excluding* proxy rights.

Rights are granted independently of one another. This means, for example, that a user who is granted add rights can create an entry but cannot delete it if delete rights have not been specifically granted. Therefore, when planning the access control policy for your directory, you must ensure that you grant rights in a way that makes sense for users. For example, it does not usually make sense to grant write permission without granting read and search permissions.



NOTE

The proxy mechanism is very powerful and must be used sparingly. Proxy rights are granted within the scope of the ACL, and there is no way to restrict who an entry that has the proxy right can impersonate; that is, when you grant a user proxy rights, that user has the ability to proxy for any user under the target; there is no way to restrict the proxy rights to only certain users. For example, if an entity has proxy rights to the **dc=example,dc=com** tree, that entity can do anything. Make sure you set the proxy ACI at the lowest possible level of the DIT; see Section 13.9.12, "Proxied Authorization ACI Example".

13.3.3.3. Rights Required for LDAP Operations

This section describes the rights you need to grant to users depending on the type of LDAP operation you want to authorize them to perform.

- Adding an entry:
  - Grant add permission on the entry being added.
  - Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the **targetattrfilters** keyword.
- Deleting an entry:
  - Grant delete permission on the entry to be deleted.
  - Grant write permission on the value of each attribute in the entry. This right is granted by default but could be restricted using the **targetattrfilters** keyword.
- Modifying an attribute in an entry:
  - Grant write permission on the attribute type.
  - Grant write permission on the value of each attribute type. This right is granted by default but could be restricted using the **targetattrfilters** keyword.
- Modifying the RDN of an entry:
  - Grant write permission on the entry.
  - Grant write permission on the attribute type used in the new RDN.
  - Grant write permission on the attribute type used in the old RDN, if you want to grant the right to delete the old RDN.

- Grant write permission on the value of attribute type used in the new RDN. This right is granted by default but could be restricted using the **targetattrfilters** keyword.

- Comparing the value of an attribute:
  - Grant compare permission on the attribute type.

- Searching for entries:
  - Grant search permission on each attribute type used in the search filter.
  - Grant read permission on attribute types used in the entry.

The permissions granted on individual attributes or entries can affect a broad range of actions; for example, there are several different permissions users must have to search the directory like the following **ldapsearch** operation:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b
"uid=bkolics,dc=example,dc=com" objectclass=* mail
```

The following ACI is used to determine whether user **bkolics** can be granted access:

```
aci: (targetattr = "mail")(version 3.0; acl "self access to
   mail"; allow (read, search) userdn = "ldap:///self";)
```

The search result list is empty because this ACI does not grant access to the ***objectclass*** attribute. If you want the search operation described above to be successful, modify the ACI to allow read and search access for the ***mail*** and ***objectclass*** attributes.

```
aci: (targetattr = "mail || objectclass")(version 3.0; acl "self
   access to mail"; allow (read, search) userdn = "ldap:///self";)
```

13.3.3.4. Permissions Syntax

In an ACI statement, the syntax for permissions is **allow|deny (***rights***)**. *rights* is a list of 1 to 8 comma-separated keywords enclosed within parentheses. Valid keywords are **read**, **write**, **add**, **delete**, **search**, **compare**, **selfwrite**, **proxy**, or **all**.

In the following example, read, search, and compare access is allowed, provided the bind rule is evaluated to be true:

```
aci: (target="ldap:///dc=example,dc=com") (version 3.0;acl "example";
    allow (read, search, compare) bind_rule;)
```

13.3.3.5. Access Control and the modrdn Operation

To explicitly deny **modrdn** rights using ACIs, target the relevant entries but omit the **targetattr** keyword. For example, to prevent the **cn=helpDeskGroup,ou=groups,dc=example,dc=com** group from renaming any entries in the set specified by the pattern **cn=*,ou=people,dc=example,dc=com**, add the following ACI:

```
aci: (target="ldap:///cn=*,ou=people,dc=example,dc=com")
    (version 3.0; acl "Deny modrdn rights to the helpDeskGroup";
    deny(write) groupdn="ldap:///cn=helpDeskGroup,ou=groups,dc=example,dc=com";)
```

13.4. BIND RULES

Depending on the ACIs defined for the directory, for certain operations, you need to *bind* to the directory. *Binding* means logging in or authenticating yourself to the directory by providing credentials (for example, a keytab for SASL or a client certificate for SSL). The credentials provided in the bind operation and the circumstances of the bind determine whether access to the directory is allowed or denied.

Every permission set in an ACI has a corresponding bind rule that details the required credentials and bind parameters.

Bind rules can be simple, such as stating that the person accessing the directory must belong to a specific group. Bind rules can also be more complex, such as requiring that a person must belong to a specific group, must log in from a machine with a specific IP address, and is restricted to access between 8 a.m. and 5 p.m.

Bind rules define who can access the directory, when, and from where by defining any of the following:

- Users, groups, and roles that are granted access.
- Locations from which an entity must bind.
- Times or days on which binding must occur.
- Types of authentication that must be in use during binding.

Additionally, bind rules can be complex constructions that combine these criteria by using Boolean operators. See Section 13.4.11, "Using Boolean Bind Rules" for more information.

13.4.1. Bind Rule Syntax

Whether access is allowed or denied depends on whether an ACI's bind rule is evaluated to be true. Bind rules use one of the two following patterns:

> *keyword* = "*expression*"; or
> *keyword* != "*expression*";

Equal (**=**) indicates that *keyword* and *expression* must match in order for the bind rule to be true, and not equal (**!=**) indicates that *keyword* and *expression* must not match in order for the bind rule to be true.

NOTE

The **timeofday** keyword also supports the inequality expressions (<, <=, >,>=). This is the only keyword that supports these expressions.

The quotation marks (**""**) around *expression* and the delimiting semicolon (**;**) are required. The expressions you can use depend on the associated *keyword*.

Table 13.3, "LDIF Bind Rule Keywords" lists each keyword and the associated expressions and indicates whether wildcard characters are allowed in the expression.

Table 13.3. LDIF Bind Rule Keywords

Keyword	Valid Expressions	Wildcard Allowed
userdn	ldap:///*distinguished_name* ldap:///all ldap:///anyone ldap:///self ldap:///parent ldap:///*suffix*??scope?(*filter*)	Yes, in DN only
groupdn	ldap:///*DN*|| *DN* ldap:///*suffix*??scope?(*filter*)	No
roledn	ldap:///*DN*|| *DN*	No
userattr	*attribute#bindType* or*attribute#value*	No
ip	*IP_address*	Yes
dns	*DNS_host_name*	Yes
dayofweek	sun mon tue wed thu fri sat	No

Keyword	Valid Expressions	Wildcard Allowed
timeofday	0 - 2359	No
authmethod	none simple ssl sasl *sasl_mechanism*	No

13.4.2. Defining User Access - userdn Keyword

User access is defined using the **userdn** keyword. The **userdn** keyword requires one or more valid distinguished names in the following format:

```
userdn = "ldap:///dn [|| ldap:///dn]...[||ldap:///dn]"
```

dn can be a DN or one of the expressions **anyone**, **all**, **self**, or **parent**:

```
userdn = "ldap:///anyone" Defines anonymous access
userdn = "ldap:///all" Defines general access
userdn =ldap:///self" Defines self access
userdn =ldap:///parent" Defines access for the parent entry
```

The **userdn** keyword can also be expressed as an LDAP filter:

```
ldap:///suffix??scope?(filter)
```

NOTE

If a DN contains a comma, the comma must be preceded by a backslash (\) escape character.

13.4.2.1. Anonymous Access (anyone Keyword)

Granting anonymous access to the directory means that anyone can access it without providing a bind DN or password and regardless of the circumstances of the bind. You can limit anonymous access to specific types of access (for example, read or search access) or to specific subtrees or individual entries within the directory.

From the Directory Server Console, you define anonymous access through the **Access Control Editor**. See Section 13.5, “Creating ACIs from the Console” .

13.4.2.2. General Access (all Keyword)

You can use bind rules to indicate that a permission applies to anyone who has successfully bound to the directory; that is, all authenticated users. This allows general access while preventing anonymous access.

From the Directory Server Console, you define general access on the **Access Control Editor**. For more information, see Section 13.5, “Creating ACIs from the Console” .

13.4.2.3. Self Access (self Keyword)

Specifies that users are granted or denied access to their own entries. In this case, access is granted or denied if the bind DN matches the DN of the targeted entry.

Self access can be configured in the **Access Control Editor** in the Directory Server Console. For more information, see Section 13.5, “Creating ACIs from the Console” .

13.4.2.4. Parent Access (parent Keyword)

Specifies that users are granted or denied access to the entry only if their bind DN is the parent of the targeted entry.

You cannot set up parent access control using the Directory Server Console.

13.4.2.5. LDAP URLs

You can dynamically target users in ACIs using a URL with an LDAP filter:

```
userdn = "ldap:///suffix??scope?(filter)"
```

For example, all users in the accounting and engineering branches of the **example** tree would be granted or denied access to the targeted resource dynamically based on the following URL:

```
userdn = "ldap:///dc=example,dc=com??sub?(|(ou=engineering)(ou=accounting))"
```

NOTE

Do not specify a host name or port number within the LDAP URL. LDAP URLs always apply to the local server.

It is possible to string multiple LDAP URLs together so that the bind user must match both filter A *and* filter B. This is done by using multiple **userdn** keyword definitions. For example:

```
userdn="ldap:///dc=example,dc=com??sub?(ou=engineering)" and userdn="ldap:///dc=example,dc=com??sub?
(manager="uid=bjensen,ou=managers,dc=example,dc=com")"
```

Using a connector such as **&&** is not allowed. For example, this is not an acceptable bind rule:

```
groupdn="ldap:///dc=example,dc=com??sub?(ou=engineering) && ldap:///dc=example,dc=com??sub?
(manager="uid=bjensen,ou=managers,dc=example,dc=com")"
```

For more information about LDAP URLs, see Appendix C, *LDAP URLs*.

13.4.2.6. Wildcards

You can also specify a set of users by using the wildcard character (*). For example, specifying a user DN of **uid=u*,dc=example,dc=com** indicates that only users with a bind DN beginning with the letter **u** are allowed or denied access based on the permissions you set.

From the Directory Server Console, you set user access from the **Access Control Editor**. For more information, see Section 13.5, “Creating ACIs from the Console” .

13.4.2.7. Examples

Table 13.4. userdn Keyword Examples

Scenario	Example	Description
Userdn keyword containing an LDAP URL	userdn = "ldap:///uid=*,dc=example,dc=com";	The bind rule is evaluated to be true if the user binds to the directory using any distinguished name of the specified pattern. For example, both of the following bind DNs would be evaluated to be true: uid=ssarette,dc=example,dc=com uid=tjaz,ou=Accounting,dc=example,dc=com This bind DN would be evaluated to be false: cn=Babs Jensen,dc=example,dc=com

Scenario	Example	Description
Userdn keyword containing logical OR of LDAP URLs	userdn="ldap:///uid=bj,dc=example,dc=com || ldap:///uid=kc,dc=example,dc=com";	The bind rule is evaluated to be true if the client binds as either of the two supplied distinguished names.
Userdn keyword excluding a specific LDAP URL	userdn != "ldap:///uid=*,ou=Accounting,dc=example,dc=com";	The bind rule is evaluated to be true if the client is not binding as a UID-based distinguished name in the accounting subtree. This bind rule only makes sense if the targeted entry is not under the accounting branch of the directory tree.
Userdn keyword containing self keyword	userdn = "ldap:///self";	The bind rule is evaluated to be true if the user is accessing the entry represented by the DN with which the user bound to the directory. That is, if the user has bound as **uid=ssarette**, **dc=example,dc=com** and the user is attempting an operation on the **uid=ssarette,dc=example,dc=com** entry, then the bind rule is true. If you want to grant all users in the **example** tree write access to their ***userPassword*** attribute, you would create the following ACI on the **dc=example,dc=com** node. `aci: (targetattr = "userPassword") (version 3.0; acl "write-self"; allow (write) userdn = "ldap:///self";)`
Userdn keyword containing the all keyword	userdn = "ldap:///all";	The bind rule is evaluated to be true for any valid bind DN. To be true, a valid distinguished name must be presented by the user for a successful bind operation. For example, if you want to grant read access to the entire tree to all authenticated users, you would create the following ACI on the **dc=example,dc=com** node: `aci:(version 3.0; acl "all-read"; allow (read) userdn="ldap:///all";)`
Userdn keyword containing the anyone keyword	userdn = "ldap:///anyone";	The bind rule is evaluated to be true for anyone; use this keyword to provide anonymous access to your directory. For example, if you want to allow anonymous read and search access to the entire **example** tree, you would create the following ACI on the **dc=example,dc=com** node: `aci: (version 3.0; acl "anonymous-read-search"; allow (read,search) userdn = "ldap:///anyone";)`

Scenario	Example	Description
Userdn keyword containing the parent keyword	userdn = "ldap:///parent";	The bind rule is evaluated to be true if the bind DN is the parent of the targeted entry. For example, if you want to grant write access to every user's child entries, you would create the following ACI on the **dc=example,dc=com** node: aci:(version 3.0; acl "parent access"; allow (write) userdn="ldap:///parent";)

13.4.3. Defining Group Access - groupdn Keyword

Members of a specific group can access a targeted resource. This is known as *group access*. Group access is defined using the **groupdn** keyword to specify that access to a targeted entry is granted or denied if the user binds using a DN that belongs to a specific group.

Group membership can be determined based on the user's DN or by using an LDAP filter to search for group members.

The **groupdn** keyword requires one or more valid distinguished names in the following format:

```
groupdn="ldap:///dn [|| ldap:///dn]...[|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the named group.

NOTE

If a DN contains a comma, the comma must be escaped by a backslash (\).

The **groupdn** keyword can also be expressed with an LDAP filter:

```
groupdn="ldap:///suffix??scope?(filter)
```

With more complex **groupdn** syntax, the value of the **groupdn** expression is a single LDAP URL. Multiple **groupdn**s can be grouped together within parentheses and use **or** or **and** connectors to define additional conditions on the group membership. For example:

```
(groupdn = "ldap:///ou=Groups,dc=example,dc=com??sub?(cn=*s_0)" or groupdn =
"ldap:///ou=Groups,dc=example,dc=com??sub?(cn=*s_1)") and groupdn =
"ldap:///ou=Groups,dc=example,dc=com??sub?(cn=*s_2)"
```

When stringing multiple **groupdn** URLs together, the keyword supports pipes to separate the URLs:

```
groupdn = "LDAPURI0 || LDAPURL1 || LDAPURL2"
```

However, it is not permissible to use ampersands (**&**), like **groupdn = "LDAPURI0 && LDAPURL1"**, or double quotes.

For example, to use two **groupdn** keywords so that the bind user must belong to both an Administrators group *and* a Managers group:

```
groupdn="ldap:///dc=example,dc=com??sub?(cn=*Administrators)" and groupdn="ldap:///dc=example,dc=com??
sub?(cn=*Managers)"
```

For more information about LDAP URLs, see Appendix C, *LDAP URLs*.

The Directory Server Console defines specific groups through the **Access Control Editor**. For more information, see Section 13.5, "Creating ACIs from the Console" .

Table 13.5. groupdn Examples

Scenario	Example	Description

Scenario	Example	Description
Groupdn keyword containing an LDAP URL	groupdn = "ldap:///cn=Administrators,dc=example,dc=com";	The bind rule is evaluated to be true if the bind DN belongs to the Administrators group. If you wanted to grant the Administrators group permission to write to the entire directory tree, you would create the following ACI on the **dc=example,dc=com** node: aci: (targetattr=*)(version 3.0; acl "Administrators-write"; allow (write) groupdn="ldap:///cn=Administrators,dc=example,dc=com";)
Groupdn keyword containing an LDAP URL with a filter	groupdn = "ldap:///dc=example,dc=com??sub?(cn=*Administrators)";	The bind rule is evaluated to be true if the bind DN belongs to any of the groups which are returned, meaning they match the filter.
Groupdn keyword containing logical OR of LDAP URLs	groupdn = "ldap:///cn=Administrators,dc=example,dc=com" || "ldap:///cn=Mail Administrators,dc=example,dc=com";	The bind rule is evaluated to be true if the bind DN belongs to either the **Administrators** or the **Mail Administrators** group.

13.4.4. Defining Role Access - roledn Keyword

Members of a specific role can access a targeted resource. This is known as *role access*. Role access is defined using the **roledn** keyword to specify that access to a targeted entry is granted or denied if the user binds using a DN that belongs to a specific role.

The **roledn** keyword requires one or more valid distinguished names in the following format:

```
roledn = "ldap:///dn [|| ldap:///dn]... [|| ldap:///dn]"
```

The bind rule is evaluated to be true if the bind DN belongs to the specified role.

NOTE

If a DN contains a comma, the comma must be escaped by a backslash (\).

The **roledn** keyword has the same syntax and is used in the same way as the **groupdn** keyword, with the exception of the LDAP filter, which is not implemented for role membership.

13.4.5. Defining Access Based on Value Matching

You can set bind rules to specify that an attribute value of the entry used to bind to the directory must match an attribute value of the targeted entry.

For example, you can specify that the bind DN must match the DN in the ***manager*** attribute of a user entry in order for the ACI to apply. In this case, only the user's manager would have access to the entry.

This example is based on DN matching. However, you can match any attribute of the entry used in the bind with the targeted entry. For example, you could create an ACI that allowed any user whose ***favoriteDrink*** attribute is **beer** to read all the entries of other users that have the same value for ***favoriteDrink***.

13.4.5.1. Using the userattr Keyword

The **userattr** keyword can be used to specify which attribute values must match between the entry used to bind and the targeted entry. You can specify any of the following:

- A user DN
- A group DN

- A role DN
- An LDAP filter, in an LDAP URL
- Any attribute type

The LDIF syntax of the **userattr** keyword is as follows:

```
userattr = "attrName#bindType
```

Using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter has the following format:

```
userattr = "attrName#attrValue
```

- *attrName* is the name of the attribute used for value matching.
- *bindType* is either **USERDN**, **GROUPDN**, or **LDAPURL**.
- *attrValue* is any string representing an attribute value.

13.4.5.1.1. Example with USERDN Bind Type

The following associates the **userattr** keyword with a bind based on the user DN:

```
userattr = "manager#USERDN"
```

The bind rule is evaluated to be true if the bind DN matches the value of the ***manager*** attribute in the targeted entry. You can use this to allow a user's manager to modify employees' attributes. This mechanism only works if the ***manager*** attribute in the targeted entry is expressed as a full DN.

The following example grants a manager full access to his or her employees' entries:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)
   (version 3.0; acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

13.4.5.1.2. Example with GROUPDN Bind Type

The following associates the **userattr** keyword with a bind based on a group DN:

```
userattr = "owner#GROUPDN"
```

The bind rule is evaluated to be true if the bind DN is a member of the group specified in the ***owner*** attribute of the targeted entry. For example, you can use this mechanism to allow a group to manage employees' status information. You can use an attribute other than ***owner*** as long as the attribute you use contains the DN of a group entry.

The group you point to can be a dynamic group, and the DN of the group can be under any suffix in the database. However, the evaluation of this type of ACI by the server is very resource intensive.

If you are using static groups that are under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?owner#GROUPDN"
```

In this example, the group entry is under the **dc=example,dc=com** suffix. The server can process this type of syntax more quickly than the previous example.

(By default, **owner** is not an allowed entry in a user's entry. You would have to extend your schema to allow this attribute in a **person** object.)

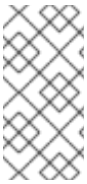
13.4.5.1.3. Example with ROLEDN Bind Type

The following associates the **userattr** keyword with a bind based on a role DN:

```
userattr = "exampleEmployeeReportsTo#ROLEDN"
```

The bind rule is evaluated to be true if the bind DN belongs to the role specified in the ***exampleEmployeeReportsTo*** attribute of the targeted entry. For example, if you create a nested role for all managers in your company, you can use

this mechanism to grant managers at all levels access to information about employees that are at a lower grade than themselves.



NOTE

This example assumes that you have added the ***exampleEmployeeReportsToattribute*** to the schema and that all employee entries contain this attribute. It also assumes that the value of this attribute is the DN of a role entry. For information on adding attributes to the schema, see Section 8.4.2, "Creating Attributes".

The DN of the role can be under any suffix in the database. If you are also using filtered roles, the evaluation of this type of ACI uses a lot of resources on the server.

If you are using a static role definition and the role entry is under the same suffix as the targeted entry, you can use the following expression:

```
userattr = "ldap:///dc=example,dc=com?employeeReportsTo#ROLEDN"
```

In this example, the role entry is under the **dc=example,dc=com** suffix. The server can process this type of syntax more quickly than the previous example.

13.4.5.1.4. Example with LDAPURL Bind Type

The following associates the **userattr** keyword with a bind based on an LDAP filter:

```
userattr = "myfilter#LDAPURL
```

The bind rule is evaluated to be true if the bind DN matches the filter specified in the *myfilter* attribute of the targeted entry. The *myfilter* attribute can be replaced by any attribute that contains an LDAP filter.

13.4.5.1.5. Example with Any Attribute Value

The following associates the **userattr** keyword with a bind based on any attribute value:

```
userattr = "favoriteDrink#Beer"
```

The bind rule is evaluated to be true if the bind DN and the target DN include the ***favoriteDrink*** attribute with a value of **Beer**.

13.4.5.1.6. Using the userattr Keyword with Inheritance

When you use the **userattr** keyword to associate the entry used to bind with the target entry, the ACI applies only to the target specified and not to the entries below it. In some circumstances, you might want to extend the application of the ACI several levels below the targeted entry. This is possible by using the parent keyword and specifying the number of levels below the target that should inherit the ACI.

When you use the **userattr** keyword in association with the **parent** keyword, the syntax is as follows:

```
userattr = "parent[inheritance_level].attrName#bindType
```

Using an attribute type that requires a value other than a user DN, group DN, role DN, or an LDAP filter, the syntax is as follows:

```
userattr = "parent[inheritance_level].attrName#attrValue
```

- *inheritance_level* is a comma-separated list that indicates how many levels below the target inherits the ACI. You can include five levels (**0**, **1**, **2**, **3**, **4**) below the targeted entry; zero (**0**) indicates the targeted entry.
- *attribute* is the attribute targeted by the **userattr** or **groupattr** keyword.
- *bindType* can be one of **USERDN**, **GROUPDN**, or **LDAPURL**.

For example:

```
userattr = "parent[0,1].manager#USERDN"
```

This bind rule is evaluated to be true if the bind DN matches the manager attribute of the targeted entry. The permissions granted when the bind rule is evaluated to be true apply to the target entry *and* to all entries immediately below it.

The example in Figure 13.1, "Using Inheritance With the userattr Keyword" indicates that user **bjensen** is allowed to read and search the **cn=Profiles** entry as well as the first level of child entries which includes **cn=mail** and **cn=news**, thus allowing her to search through her own mail and news IDs.

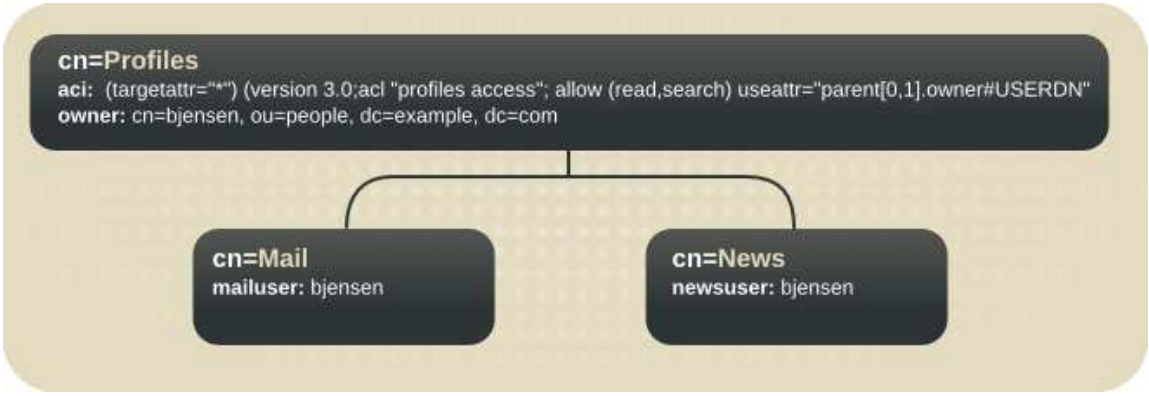


Figure 13.1. Using Inheritance With the userattr Keyword

In this example, if you did not use inheritance, you would have to do one of the following to achieve the same result:

- Explicitly set read and search access for user **bjensen** on the **cn=Profiles**, **cn=mail**, and **cn=news** entries in the directory.
- Add the owner attribute with a value of **bjensen** to the **cn=mail** and **cn=news** entries, and then add the following ACI to the **cn=mail** and **cn=news** entries.

```
aci: (targetattr="*") (version 3.0; acl "profiles access"; allow (read,search)
    userattr="owner#USERDN";)
```

13.4.5.1.7. Granting Add Permission Using the userattr Keyword

Using the **userattr** keyword in conjunction with **all** or **add** permissions does not behave as one would typically expect. Typically, when a new entry is created in the directory, Directory Server evaluates access rights on the entry being created and not on the parent entry. However, in the case of ACIs using the **userattr** keyword, this behavior could create a security hole, and the server's normal behavior is modified to avoid it.

Consider the following example:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*) (version 3.0;
    acl "manager-write"; allow (all) userattr = "manager#USERDN";)
```

This ACI grants managers all rights on the entries of employees that report to them. However, because access rights are evaluated on the entry being created, this type of ACI would also allow any employee to create an entry in which the manager attribute is set to their own DN. For example, disgruntled employee Joe (**cn=Joe,ou=eng,dc=example,dc=com**) might want to create an entry in the Human Resources branch of the tree to use (or misuse) the privileges granted to Human Resources employees.

He could do this by creating the following entry:

```
dn: cn= Trojan Horse,ou=Human Resources,dc=example,dc=com
objectclass: top
...
cn: Trojan Horse
manager: cn=Joe,ou=eng,dc=example,dc=com
```

To avoid this type of security threat, the ACI evaluation process does not grant add permission at level 0, to the entry itself. You can, however, use the **parent** keyword to grant add rights below existing entries. You must specify the number of levels below the parent for add rights. For example, the following ACI allows child entries to be added to any entry in the **dc=example,dc=com** that has a ***manager*** attribute that matches the bind DN:

```
aci: (target="ldap:///dc=example,dc=com")(targetattr=*)
    (version 3.0; acl "parent-access"; allow (add)
    userattr = "parent[0,1].manager#USERDN";)
```

This ACI ensures that add permission is granted only to users whose bind DN matches the manager attribute of the parent entry.

13.4.6. Defining Access from a Specific IP Address

NOTE

Directory Server supports both IPv4 and IPv6 IP addresses.

Using bind rules, you can indicate that the bind operation must originate from a specific IP address. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on an IP address is as follows:

```
ip = "IP_address" or ip != "IP_address"
```

The IP address must be expressed in dot notation. You can use the wildcard character (*) to include multiple machines. For example, the following string is valid:

```
ip = "12.123.1.*";
```

The bind rule is evaluated to be true if the client accessing the directory is located at the named IP address. This can be useful for allowing certain kinds of directory access only from a specific subnet or machine.

For example, use a wildcard IP address such as **12.3.45.*** to specify a specific subnetwork or **123.45.6.*+255.255.255.115** to specify a subnetwork mask.

From the Directory Server Console, you can define specific machines to which the ACI applies through the **Access Control Editor**. For more information, see Section 13.5, “Creating ACIs from the Console” .

13.4.7. Defining Access from a Specific Domain

A bind rule can specify that the bind operation must originate from a particular domain or host machine. This is often used to force all directory updates to occur from a given machine or network domain.

The LDIF syntax for setting a bind rule based on the DNS host name is as follows:

```
dns = "DNS_Hostname or dns != "DNS_Hostname
```

WARNING

The **dns** keyword requires that the naming service used on your machine is DNS. If the name service is not DNS, use the **ip** keyword instead.

The **dns** keyword requires a fully qualified DNS domain name. Granting access to a host without specifying the domain creates a potential security threat. For example, the following expression is allowed but not recommended:

```
dns = "legend.eng";
```

Instead, use a fully qualified name:

```
dns = "legend.eng.example.com";
```

The **dns** keyword allows wildcards. For example:

```
dns = "*.example.com";
```

The bind rule is evaluated to be true if the client accessing the directory is located in the named domain. This can be useful for allowing access only from a specific domain. Wildcards will not work if your system uses a naming service other than DNS. In such a case, if you want to restrict access to a particular domain, use the **ip** keyword, as described in Section 13.4.6, “Defining Access from a Specific IP Address” .

13.4.8. Requiring a Certain Level of Security in Connections

A bind rule can specify that an operation must occur over a connection with a certain level of security. This forces operations — such as password change operations — to be performed over an SSL/TLS, Start TLS, or SASL connection. The security of the connection is determined by its *security strength factor*, which sets the minimum key strength required to process operations. The value for the SSF for any operation is the higher of the values between an SSL/TLS connection and a SASL bind; this means that if a server is configured to run over TLS and a replication agreement is configured for SASL/GSSAPI, the SSF for the operation is whichever available encryption type is more secure.

The SSF can be specified using any comparison pattern, like equals (=), less-than and greater-than, (< and >), is-not (!), less-than-or-equal-to (<=), or greater-than-or-equal-to (>=). The LDIF syntax for setting a bind rule based on the SSF is as follows:

```
ssf = "key_strength"
ssf >= "key_strength"
```

The **ssf** keyword accepts any positive whole number. If this is set to 0, than no secure connection is required for an operation.

The bind rule is evaluated to be true if the client accesses the directory using a connection of adequate strength.

13.4.9. Defining Access at a Specific Time of Day or Day of Week

You can use bind rules to specify that binding can only occur at a certain time of day or on a certain day of the week. For example, you can set a rule that allows access only if it is between the hours of 8 a.m. and 5 p.m. Monday through Friday. The time used to evaluate access rights is the time on the Directory Server, not the time on the client.

The LDIF syntax for setting a bind rule based on the time of day is as follows:

```
timeofday operator time
```

operator can be one of the following symbols:

equal to (=)
not equal to (!=)
greater than (>)
greater than or equal to (>=)
less than (<)
less than or equal to (<=)

The **timeofday** keyword requires a time of day expressed in hours and minutes in the 24 hour clock (0 to 2359).

NOTE

The time on the Directory Server is used for the evaluation, not the time on the client.

The LDIF syntax for setting a bind rule based on the day in the week is as follows:

```
dayofweek = "day1, day2 ...
```

The possible values for the **dayofweek** keyword are the English three-letter abbreviations for the days of the week: **sun**, **mon**, **tue**, **wed**, **thu**, **fri**, **sat**.

13.4.9.1. Examples

The following are examples of the **timeofday** and **dayofweek** syntax:

- The bind rule is evaluated to be true if the client is accessing the directory at noon.

```
timeofday = "1200";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time other than 1 a.m.

```
timeofday != "0100";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time after 8 a.m.

```
timeofday > "0800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at any time before 6 p.m.

```
timeofday < "1800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at 8 a.m. or later.

```
timeofday >= "0800";
```

- The bind rule is evaluated to be true if the client is accessing the directory at 6 p.m. or earlier.

```
timeofday <= "1800";
```

- The bind rule is evaluated to be true if the client is accessing the directory on Sunday, Monday, or Tuesday.

```
dayofweek = "Sun, Mon, Tue";
```

13.4.10. Defining Access Based on Authentication Method

The **authmethod** keyword sets the specific method that a client uses to bind to the directory. There are four available authentication methods:

- *None*. Authentication is not required. This is the default. It represents anonymous access.
- *Simple*. The client must provide a user name and password to bind to the directory.
- *SSL*. The client must bind to the directory using some kind of PKI credentials, meaning a client must present an SSL certificate either in a database or on a smart card, token, or some other device.

  Certificate-based authentication, as one method, is described in Section 7.10, “Using Client (Certificate-Based) Authentication”.
- *SASL*. The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. Directory Server supports several SASL mechanisms: **PLAIN**, **EXTERNAL**, **CRAM-MD5**, **DIGEST-MD5** (for Kerberos systems), and **GSS-API** (for Kerberos systems). For information on setting up SASL, see Section 7.11, “Setting up SASL Identity Mapping” .

NOTE

You cannot set up authentication-based bind rules through the **Access Control Editor**.

The LDIF syntax for setting a bind rule based on an authentication method is as follows:

```
authmethod = "auth_mechanism
```

auth_mechanism can be **none**, **simple**, **ssl**, or **"sasl** *sasl_mechanism*".

13.4.10.1. Examples

The following are examples of the **authmethod** keyword:

- Authentication is not checked during bind rule evaluation.

```
authmethod = "none";
```

- The bind rule is evaluated to be true if the client is accessing the directory using a user name and password.

```
authmethod = "simple";
```

- The bind rule is evaluated to be true if the client authenticates to the directory using a certificate over LDAPS. This is not evaluated to be true if the client authenticates using simple authentication (bind DN and password) over LDAPS. The **authmethod = "ssl"** means that a certificate must be presented to authenticate to the server. This does *not* configure a required connection type, even though SSL has to be used with certificate-based authentication.

```
authmethod = "ssl";
```

- The bind rule is evaluated to be true if the client is accessing the directory using the SASL DIGEST-MD5 mechanism.

```
authmethod = "sasl DIGEST-MD5";
```

13.4.11. Using Boolean Bind Rules

Bind rules can be complex expressions that use the Boolean expressions **AND**, **OR**, and **NOT** to set very precise access rules. You cannot use the Directory Server Console to create Boolean bind rules. You must create an LDIF statement.

The LDIF syntax for a Boolean bind rule is as follows:

bind_rule [*boolean*][*bind_rule*][*boolean*][*bind_rule*]...;)

For example, this bind rule is evaluated to be true if the bind DN is a member of either the administrator's group or the **Mail Administrator**'s group and if the client is running from within the **example.com** domain:

```
(groupdn = "ldap:///cn=administrators,dc=example,dc=com" or
    groupdn = "ldap:///cn=mail administrators,dc=example,dc=com" and
    dns = "*.example.com";)
```

The trailing semicolon (;) is a required delimiter that must appear after the final bind rule.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.
- **NOT** before **AND** or **OR** operators.
- **OR** and **AND** operators have no order of precedence.

Consider the following Boolean bind rules:

(*bind_rule_A*) OR (*bind_rule_B*)
(*bind_rule_B*) OR (*bind_rule_A*)

Because Boolean expressions are evaluated from left to right, in the first case, bind rule A is evaluated before bind rule B, and, in the second case, bind rule B is evaluated before bind rule A.

However, the Boolean **NOT** is evaluated *before* the Boolean **OR** and Boolean **AND**. Thus, in the following example, bind rule B is evaluated before bind rule A despite the left-to-right rule.

(*bind_rule_A*) AND NOT (*bind_rule_B*)

13.5. CREATING ACIS FROM THE CONSOLE

You can use the Directory Server Console to view, create, edit, and delete access control instructions for your directory:

- Section 13.5.1, “Displaying the Access Control Editor”
- Section 13.5.2, “Creating a New ACI”
- Section 13.5.3, “Editing an ACI”

- Section 13.5.4, "Deleting an ACI"

See Section 13.9, "Access Control Usage Examples" for a collection of access control rules commonly used in Directory Server security policies, along with step-by-step instructions for using the Directory Server Console to create them.

The **Access Control Editor** prevents creating more complex ACIs in visual editing mode, especially ACIs with any of these characteristics:

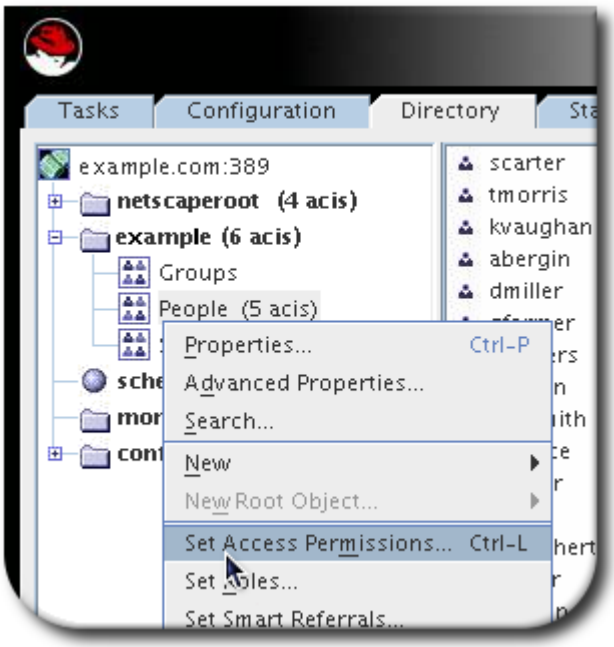
- Deny access (Section 13.3.3.4, "Permissions Syntax").
- Create value-based ACIs (Section 13.3.2.2, "Targeting Attributes").
- Define parent access (Section 13.4.2.4, "Parent Access (parent Keyword)").
- Create ACIs that contain Boolean bind rules (Section 13.4.11, "Using Boolean Bind Rules").
- Create ACIs that use the **roledn**, **userattr**, **authmethod** keywords.

NOTE

In the **Access Control Editor**, click the **Edit Manually** button at any time to check the LDIF representation of the ACI changes made through the graphical interface.

13.5.1. Displaying the Access Control Editor

1. Start the Directory Server Console. Log in using the bind DN and password of a privileged user, such as the Directory Manager, who has write access to the ACIs configured for the directory.
2. Select the **Directory** tab.
3. Right-click the entry in the navigation tree for which to set access control, and select **Set Access Permissions** from the pop-up menu.



Alternatively, highlight the entry, and select **Set Access Permissions** from the **Object** menu.

4. Click **New** to open the **Access Control Editor**.

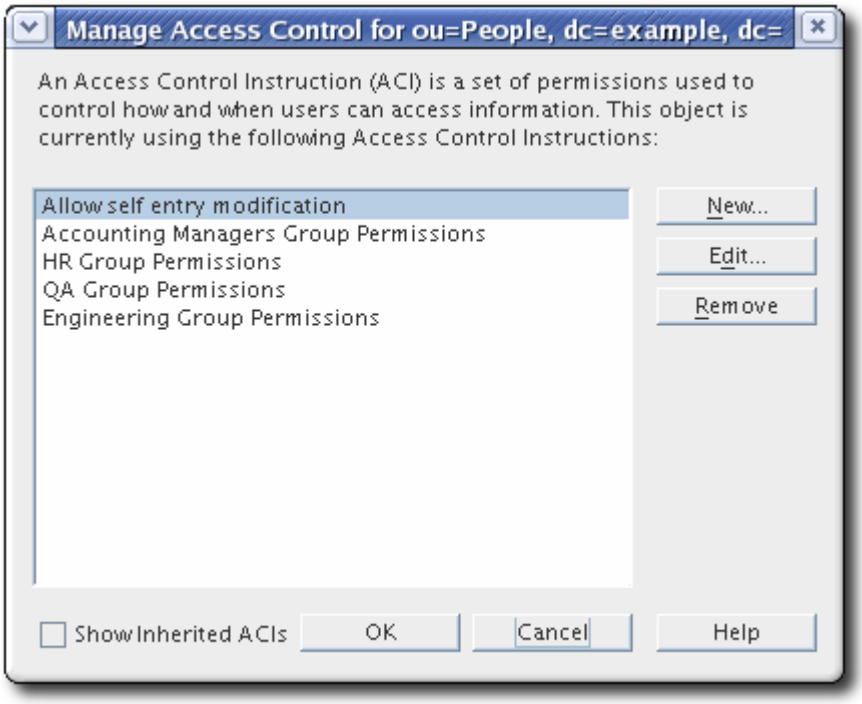


Figure 13.2. Access Control Editor Window

13.5.2. Creating a New ACI

To create a new ACI in the Directory Server Console:

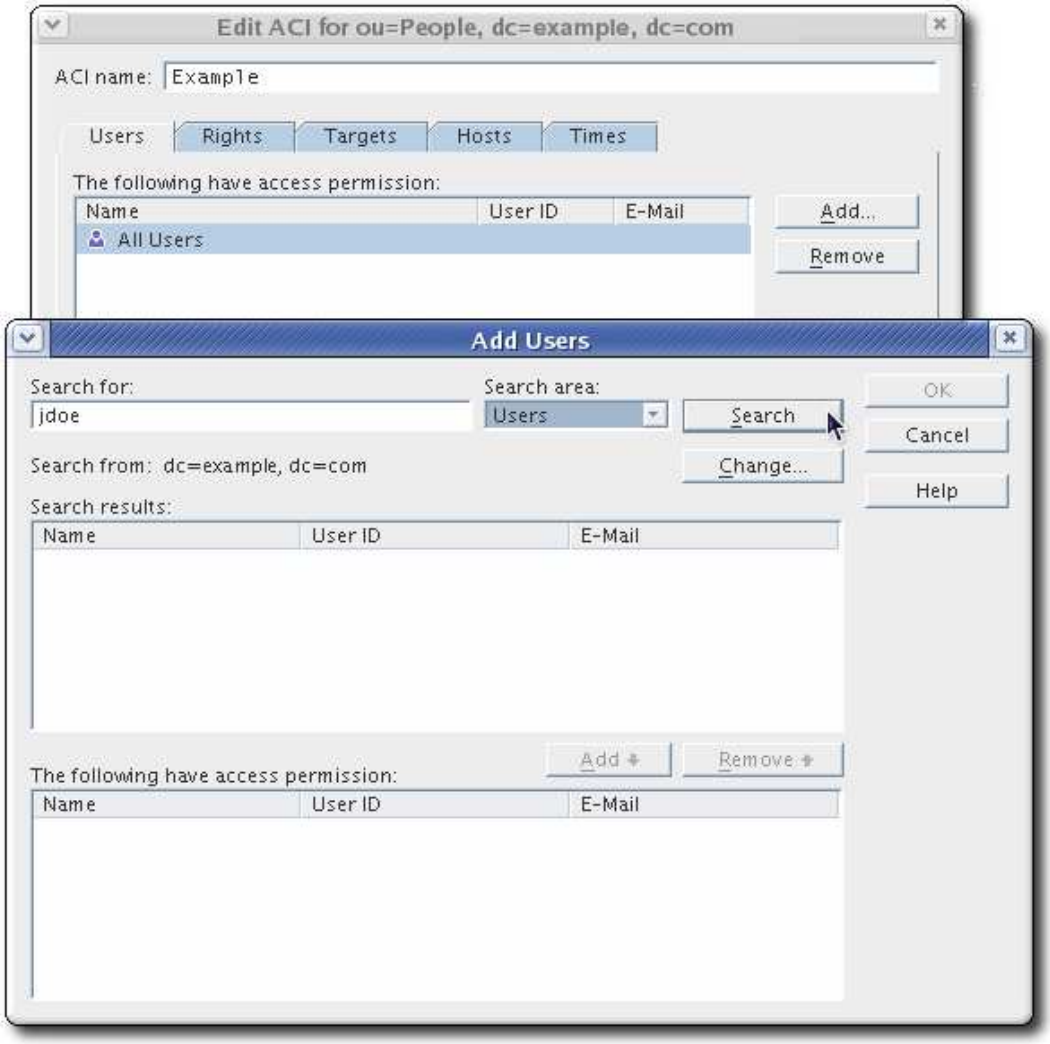
1. Open the **Access Control Editor**, as described in Section 13.5.1, "Displaying the Access Control Editor" .

   If the view displayed is different from Figure 13.2, "Access Control Editor Window" , click the **Edit Visually** button.

2. Type the ACI name in the **ACI Name** field.

   The name can be any unique string to identify the ACI. If you do not enter a name, the server uses **unnamed ACI**.

3. In the **Users/Groups** tab, select the users to whom you are granting access by highlighting **All Users** or clicking the **Add** button to search the directory for the users to add.



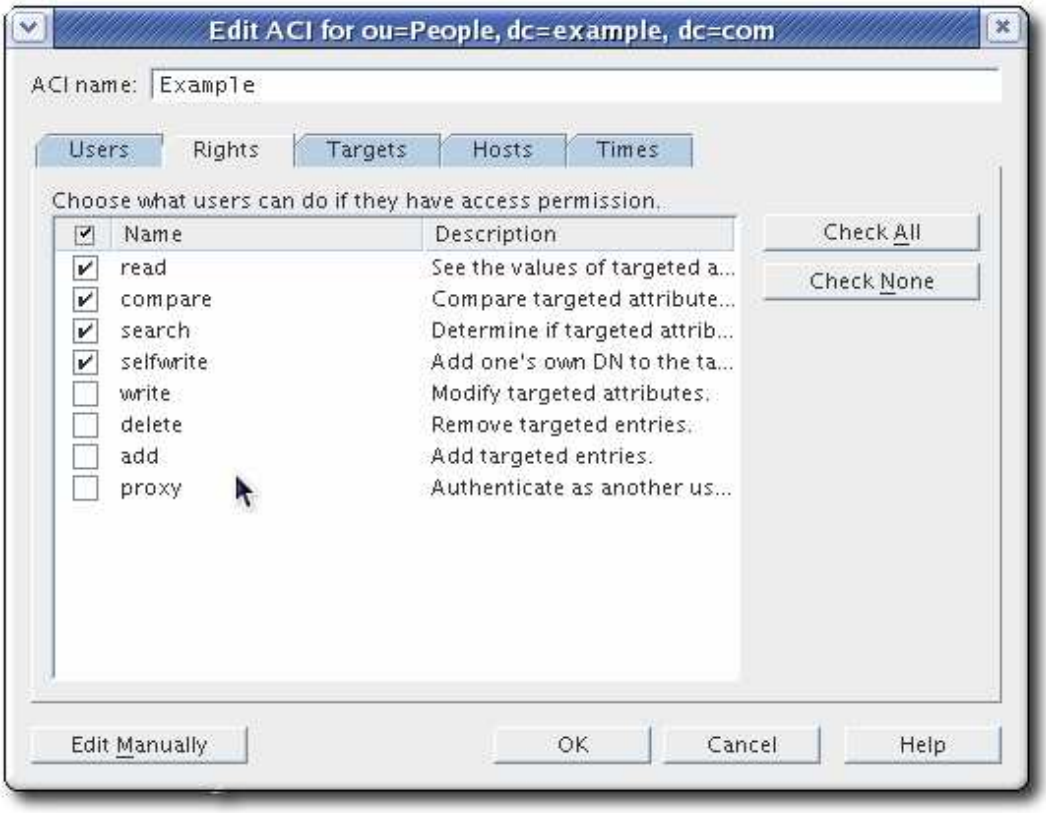
1. Select a search area from the drop-down list, enter a search string in the **Search** field, and click the **Search** button. You can use wildcards (an asterisk, *) to search for partial user names. The search results are displayed in the list below.

2. Highlight the entries you want in the search result list, and click the **Add** button to add them to the list of entries which have access permission.

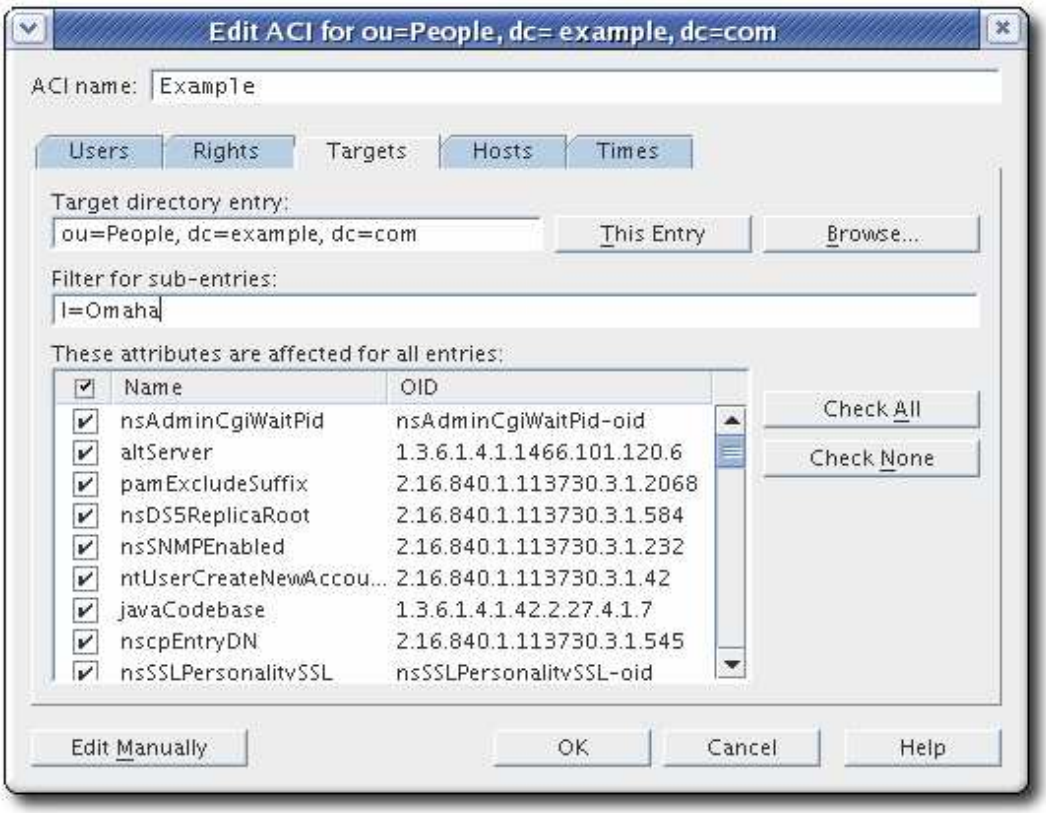
3. Click **OK** to dismiss the **Add Users and Groups** window.

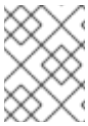
The selected entries are now listed on the **Users/Groups** tab in the ACI editor.

4. In the **Access Control Editor**, click the **Rights** tab, and use the check boxes to select the rights to grant.



5. Click the **Targets** tab. Click **This Entry** to display the current node as the target for the ACI or click **Browse** to select a different suffix.





NOTE

You can change the value of the target DN, but the new DN must be a direct or indirect child of the selected entry.

If you do not want every entry in the subtree under this node to be targeted by the ACI, enter a filter in the **Filter for Sub-entries** field. The filter applies to every entry below the target entry; for example, setting a filter of **ou=Sales** means that only entries with **ou=Sales** in their DN are returned.

Additionally, you can restrict the scope of the ACI to only certain attributes by selecting the attributes to target in the attribute list.

6. Click the **Hosts** tab, then the **Add** button to open the **Add Host Filter** dialog box.

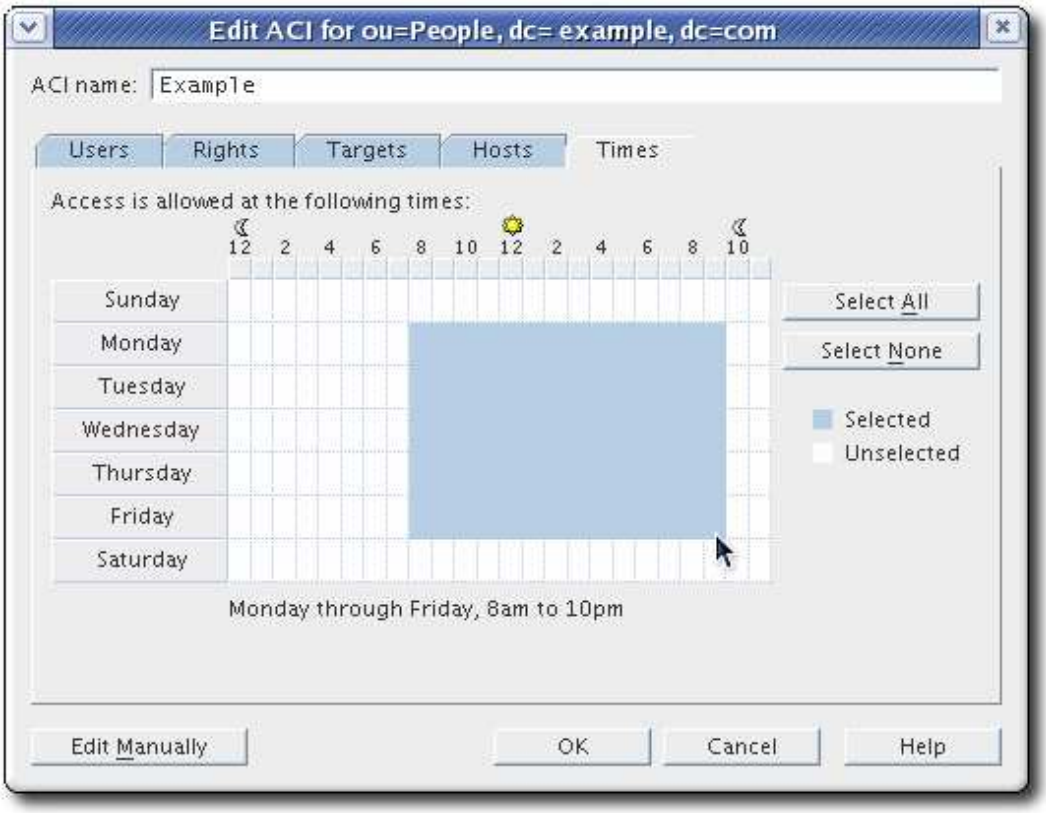
You can specify a host name or an IP address. With an IP address, you can use an asterisk (*) as a wildcard.

NOTE

Directory Server supports both IPv4 and IPv6 IP addresses.

7. Click the **Times** tab to display the table showing at what times access is allowed.

By default, access is allowed at all times. You can change the access times by clicking and dragging the cursor over the table. You cannot select discrete blocks of time, only continuous time ranges.



8. Click **OK** when all of the configuration is complete.

The **Access Control Editor** closes, and the new ACI is listed in the **Access Control Manager** window.

NOTE

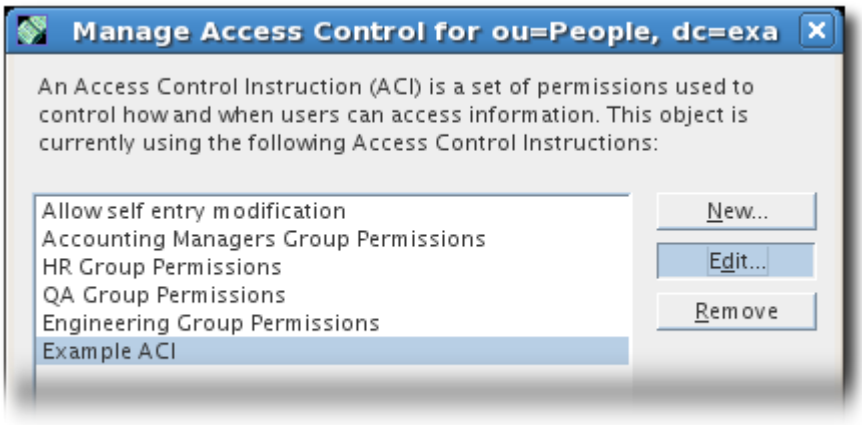
For any point of creating the ACI, click the **Edit Manually** button to display the LDIF statement corresponding to the wizard input. This statement can be edited directly, but the changes may not be visible in the graphical interface.

13.5.3. Editing an ACI

1. In the **Directory** tab, right-click the top entry in the subtree, and choose **Set Access Permissions** from the pop-up menu.

The **Access Control Manager** window opens, listing the ACIs belonging to the entry.

2. In the **Access Control Manager** window, highlight the ACI to edit, and click **Edit**.

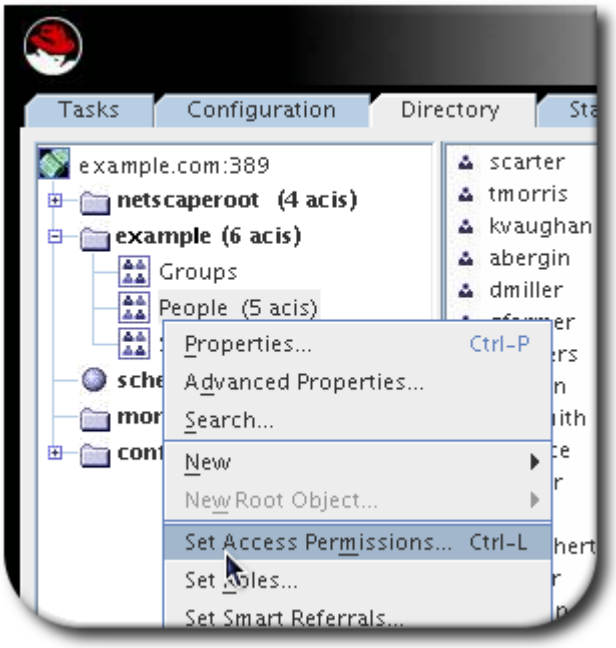


3. Make the edits to the ACI in the **Access Control Editor**; the different screens are described more in Section 13.5.2, "Creating a New ACI" and in the online help.

4. When the edits are complete, click **OK**.

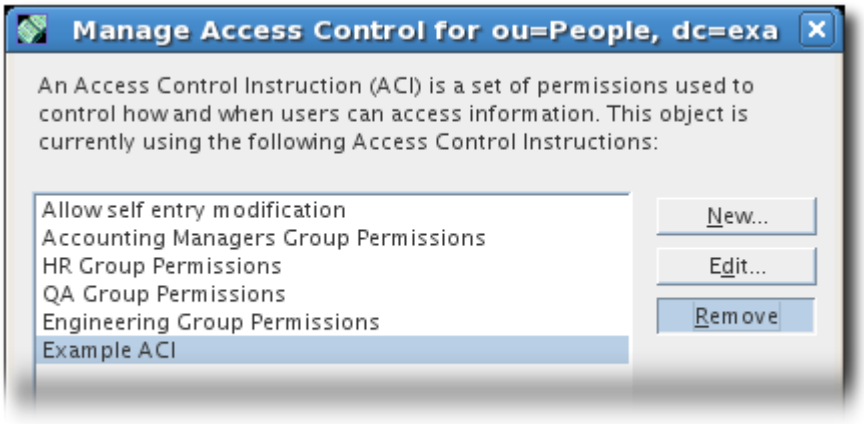
13.5.4. Deleting an ACI

1. In the **Directory** tab, right-click the top entry in the subtree, and choose **Set Access Permissions** from the pop-up menu.



2. In the **Access Control Manager** window, select the ACI to delete.

3. Click **Remove**.



The ACI is no longer listed in the **Access Control Manager** window.

13.6. VIEWING ACIS

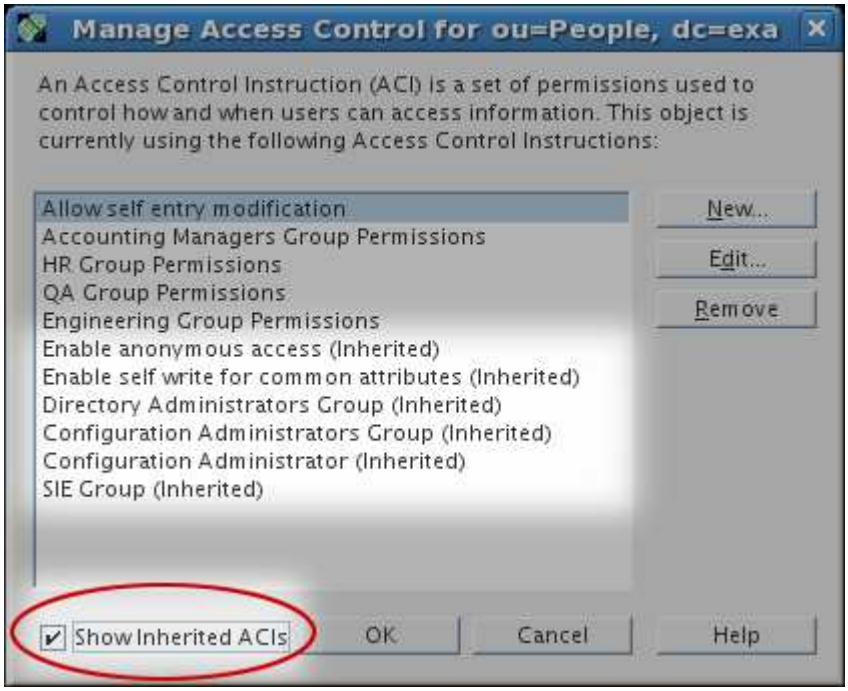
All the ACIs under a single suffix in the directory can be viewed from the command line by using the following **ldapsearch** command:

```
ldapsearch -x -D bind_dn -w password -p server_port -h server_hostname (aci=*) aci
```

See Chapter 10, *Finding Directory Entries* for information on using the **ldapsearch** utility.

From the Directory Server Console, all of the ACIs that apply to a particular entry can be viewed through the **Access Control Manager**.

1. Start the Directory Server Console.
2. In the **Directory** tab, right-click the entry in the navigation tree, and select **Set Access Permissions**.
3. Check the **Show Inherited ACIs** check box to display all ACIs created on entries above the selected entry that also apply.



13.7. CHECKING ACCESS RIGHTS ON ENTRIES (GET EFFECTIVE RIGHTS)

Finding the access rights that a user has on attributes within a specific entry offers a convenient way for administrators to find and control the access rights.

Get effective rights is a way to extend directory searches to display what access rights — such as read, search, write and self-write, add, and delete — a user has to a specified entry.

In Directory Server, regular users can check their rights over entries which they can view and can check other people's access to their personal entries. The Directory Manager can check rights that one user has over another user.

There are two common situations where checking the effective rights on an entry are useful:

- An administrator can use the get effective rights command in order to better organize access control instructions for the directory. It is frequently necessary to restrict what one group of users can view or edit versus another group. For instance, members of the **QA Managers** group may have the right to search and read attributes like ***manager*** and ***salary*** but only **HR Group** members have the rights to modify or delete them. Checking the effective rights for a user or group is one way to verify that the appropriate access controls are in place.
- A user can run the get effective rights command to see what attributes he can view or modify on his personal entry. For instance, a user should have access to attributes such as ***homePostalAddress*** and ***cn*** but may only have read access to ***manager*** and ***salary*** attributes.

There are three people involved in a get effective rights search. The first is the person running the search command, the *requester*. The rights are checked (with a variety of permutations) to see what rights Person A has over Entry B. The person whose rights are being checked (Person A) is the *GER subject*; as in, their rights are the subject of the search. The entry or entries to which the person has rights (Entry B) is the *target* of the search or the *search base*.

13.7.1. Rights Shown with a Get Effective Rights Search

Any get effective rights search, both when viewing an entry in the Directory Server Console and searching for it in the command line, shows the rights that User A has to User B's entry.

There are two kinds of access rights that can be allowed to any entry. The first are upper-level rights, *rights on the entry itself*, which means that kinds of operations that the User A can perform on User B's entry as a whole. The second level of access rights are more granular, show what *rights for a given attribute* User A has. In this case, User A may have different kinds of access permissions for different attributes in the same entry. Whatever access controls are allowed for a user are the *effective* rights over that entry.

For example:

```
entryLevelRights: vadn
attributeLevelRights: givenName:rscWO, sn:rscW, objectClass:rsc, uid:rsc, cn:rscW
```

Table 13.6, “Entry Rights” and Table 13.7, “Attribute Rights” show the access rights to entries and attributes, respectively, that are returned by a get effective rights search.

Table 13.6. Entry Rights

Permission	Description
a	Add an entry.
d	Delete this entry.
n	Rename the DN.
v	View the entry.

Table 13.7. Attribute Rights

Permission	Description
r	Read.
s	Search.
w	Write (**mod-add**).
o	Obliterate(**mod-del**). Analogous to delete.

Permission	Description
c	Compare.
W	Self-write.
O	Self-delete.

13.7.2. The Format of a Get Effective Rights Search

Get effective rights (sometimes called GER) is an extended directory search; the GER parameters are defined with the **-E** option to pass an LDAP control with the **ldapsearch** command. (If an **ldapsearch** is run without the **-E** option, then, naturally, the entry is returned as normal, without any get effective rights information.)

```
ldapsearch -x -D bind_dn -w password -p server_port -h server_hostname -E
[!]1.3.6.1.4.1.42.2.27.9.5.2=:GER_subject (searchFilter) attributeList
```

- **-b** is the base DN of the subtree or entry used to search for the GER subject.

  If the search base is a specific entry DN or if only one entry is returned, then the results show the rights the requester has over that specific entry. If multiple entries beneath the search base match the filter, then the search returns every matching entry, with the rights for the requester over each entry.

- **1.3.6.1.4.1.42.2.27.9.5.2** is the OID for the get effective rights control.

- The exclamation point (**!**) specifies whether the search operation should return an error if the server does not support this control (**!**) or if it should be ignored and let the search return as normal (nothing).

- The *GER_subject* is the person whose rights are being checked. If the *GER_subject* is left blank (**dn:**), than the rights of an anonymous user are returned.

- An optional *attributeList* limits the get effective rights results to the specified attribute or object class. As with a regular **ldapsearch**, this can give specific attributes, like ***mail***. If no attributes are listed, then every present attribute for the entry is returned. Using an asterisk (*****) returns the rights for every possible attribute for the entry, both existing attribute and non-existent attributes. Using an plus sign (**+**) returns operational attributes for the entry. Examples for checking rights for specific attributes are given in Section 13.7.3.2, "Examples of Get Effective Rights Searches for Non-Existent Attributes" and Section 13.7.3.3, "Examples of Get Effective Rights Searches for Specific Attributes or Object Classes".

The crux of a get effective rights search is the ability to check what rights the GER subject (**-E**) has to the targets of the search (**-b**). The get effective rights search is a regular **ldapsearch**, in that it simply looks for entries that match the search parameters and returns their information. The get effective rights option adds extra information to those search results, showing what rights a specific user has over those results. That GER subject user can be the requester himself (**-D** is the same as **-E**) or someone else.

If the requester is a regular user (not the Directory Manager), then the requester can only see the effective that a GER subject has on the requester's own entry. That is, if John Smith runs a request to see what effective rights Babs Jensen has, then he can only get the effective rights that Babs Jensen has on his own entry. All of the other entries return an insufficient access error for the effective rights.

There are three general scenarios for a regular user when running a get effective rights search:

- User A checks the rights that he has over other directory entries.
- User A checks the rights that he has to his personal entry.
- User A checks the rights that User B has to User A's entry.

The get effective rights search has a number of flexible different ways that it can check rights on attributes.

13.7.3. Examples of GER Searches

There are a number of different ways to run GER searches, depending on the exact type of information that needs to be returned and the types of entries and attributes being searched.

13.7.3.1. General Examples on Checking Access Rights

One common scenario for effective rights searches is for a regular user to determine what changes he can make to his personal entry.

For example, Ted Morris wants to check the rights he has to his entry. Both the **-D** and **-E** options give his entry as the requester. Since he is checking his personal entry, the **-b** option also contains his DN.

Example 13.1. Checking Personal Rights (User A to User A)

```
ldapsearch -x -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -w secret -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=People,dc=example,dc=com
givenName: Ted
sn: Morris
ou: IT
ou: People
l: Santa Clara
manager: uid=jsmith,ou=People,dc=example,dc=com
roomNumber: 4117
mail: tmorris@example.com
facsimileTelephoneNumber: +1 408 555 5409
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: tmorris
cn: Ted Morris
userPassword: {SSHA}bz0uCmHZM5b357zwrCUCJs1IOHtMD6yqPyhxBA==
entryLevelRights: v
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc, manager:rsc, roomNumber:rscwo, mail:rscwo,
facsimileTelephoneNumber:rscwo, objectClass:rsc, uid:rsc, cn:rsc, userPassword:wo
```

Ted Morris may, for example, be a manager or work in a department where he has to edit other user's entries, such as IT or human resources. In this case, he may want to check what rights he has to another user's entry, as in Example 13.2, “Personally Checking the Rights of One User over Another (User A to User B)” , where Ted (**-D**) checks his rights (**-E**) to Dave Miller's entry (**-b**):

Example 13.2. Personally Checking the Rights of One User over Another (User A to User B)

```
ldapsearch -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -w secret -b
"uid=dmiller,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=dmiller,ou=People,dc=example,dc=com
... snip ...
entryLevelRights: vad
attributeLevelRights: givenName:rscwo, sn:rscwo, ou:rscwo, l:rscwo, manager:rsc, roomNumber:rscwo,
mail:rscwo, facsimileTelephoneNumber:rscwo, objectClass:rscwo, uid:rscwo, cn:rscwo, userPassword:rswo
```

For all attributes, Ted Morris has read, search, compare, modify, and delete permissions to Dave Miller's entry. These results are different than the ones returned in checking Ted Morris's access to his own entry, since he personally had only read, search, and compare rights to most of these attributes.

The Directory Manager has the ability to check the rights that one user has over another user's entry. In Example 13.3, “The Directory Manager's Checking the Rights of One User over Another (User A to User B)”, the Directory Manager is checking the rights that a manager, Jane Smith (**-E**), has over her subordinate, Ted Morris (**-b**):

Example 13.3. The Directory Manager's Checking the Rights of One User over Another (User A to User B)

```
ldapsearch -p 389 -h server.example.com -D "cn=directory manager" -w secret -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=jsmith,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=People,dc=example,dc=com
... snip ...
```

```
entryLevelRights: vadn
attributeLevelRights: givenName:rscwo, sn:rscwo, ou:rscwo, l:rscwo, manager:rscwo, roomNumber:rscwo,
mail:rscwo, facsimileTelephoneNumber:rscwo, objectClass:rscwo, uid:rscwo, cn:rscwo, userPassword:rscwo
```

Only an administrator can retrieve the effective rights that a different user has on an entry. If Ted Morris tried to determine Dave Miller's rights to Dave Miller's entry, then he would receive an insufficient access error:

```
ldapsearch -p 389 -h server.example.com -D "uid=dmiller,ou=people,dc=example,dc=com" -w secret -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=tmorris,ou=people,dc=example,dc=com' "(objectClass=*)"

ldap_search: Insufficient access
ldap_search: additional info: get-effective-rights: requester has no g permission on the entry
```

However, a regular user can run a get effective rights search to see what rights another user has to his personal entry. In Example 13.4, “Checking the Rights Someone Else Has to a Personal Entry” , Ted Morris checks what rights Dave Miller has on Ted Morris's entry.

Example 13.4. Checking the Rights Someone Else Has to a Personal Entry

```
ldapsearch -p 389 -h server.example.com -D "uid=tmorris,ou=people,dc=example,dc=com" -w secret -b
"uid=tmorris,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=dmiller,ou=people,dc=example,dc=com' "(objectClass=*)"

dn: uid=tmorris,ou=people,dc=example,dc=com
... snip ...
entryLevelRights: v
attributeLevelRights: givenName:rsc, sn:rsc, ou:rsc, l:rsc,manager:rsc, roomNumber:rsc, mail:rsc,
facsimileTelephoneNumber:rsc, objectClass:rsc, uid:rsc, cn:rsc, userPassword:none
```

In this case, Dave Miller has the right to view the DN of the entry and to read, search, and compare the ***ou***, ***givenName***, ***l***, and other attributes, and no rights to the ***userPassword*** attribute.

13.7.3.2. Examples of Get Effective Rights Searches for Non-Existent Attributes

By default, information is not given for attributes in an entry that do not have a value; for example, if the ***userPassword*** value is removed, then a future effective rights search on the entry above would not return any effective rights for ***userPassword***, even though self-write and self-delete rights could be allowed.

Using an asterisk (*) with the get effective rights search returns every attribute available for the entry, including attributes not set on the entry.

Example 13.5. Returning Effective Rights for Non-Existent Attributes

```
ldapsearch -D "cn=directory manager" -w secret -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "*"

dn: uid=scarter,ou=People,dc=example,dc=com
givenName: Sam
telephoneNumber: +1 408 555 4798
sn: Carter
ou: Accounting
ou: People
l: Sunnyvale
manager: uid=dmiller,ou=People,dc=example,dc=com
roomNumber: 4612
mail: scarter@example.com
facsimileTelephoneNumber: +1 408 555 9700
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
uid: scarter
cn: Sam Carter
userPassword: {SSHA}Xd9Jt8g1UsHC8enNDrEmxj3iJPKQLItlDYdD9A==
entryLevelRights: vadn
attributeLevelRights: objectClass:rscwo, aci:rscwo, sn:rscwo, cn:rscwo, description:rscwo, seeAlso:rscwo,
```

```
telephoneNumber:rscwo, userPassword:rscwo, destinationIndicator:rscwo, facsimileTelephoneNumber:rscwo,
internationaliSDNNumber:rscwo, l:rscwo, ou:rscwo, physicalDeliveryOfficeName:rscwo, postOfficeBox:rscwo,
postalAddress:rscwo, postalCode:rscwo, preferredDeliveryMethod:rscwo, registeredAddress:rscwo, st:rscwo,
street:rscwo, teletexTerminalIdentifier:rscwo, telexNumber:rscwo, title:rscwo, x121Address:rscwo, audio:rscwo,
businessCategory:rscwo, carLicense:rscwo, departmentNumber:rscwo, displayName:rscwo,
employeeType:rscwo, employeeNumber:rscwo, givenName:rscwo, homePhone:rscwo,
homePostalAddress:rscwo, initials:rscwo, jpegPhoto:rscwo, labeledUri:rscwo, manager:rscwo, mobile:rscwo,
pager:rscwo, photo:rscwo, preferredLanguage:rscwo, mail:rscwo, o:rscwo, roomNumber:rscwo,
secretary:rscwo, uid:rscwo,x500UniqueIdentifier:rscwo, userCertificate:rscwo, userSMIMECertificate:rscwo,
userPKCS12:rscwo
```

All of the attributes available for the entry, such as ***secretary***, are listed, even though that attribute is non-existent.

13.7.3.3. Examples of Get Effective Rights Searches for Specific Attributes or Object Classes

Taking the attribute-related GER searches further, it is possible to search for the rights to a specific attribute and set of attributes and to list all of the attributes available for one of the object classes set on the entry.

One of the options listed in the formatting example in Section 13.7.2, "The Format of a Get Effective Rights Search" is *attributeList*. To return the effective rights for only specific attributes, list the attributes, separated by spaces, at the end of the search command.

Example 13.6. Get Effective Rights Results for Specific Attributes

```
ldapsearch -D "cn=directory manager" -w secret -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" cn mail initials

dn: uid=scarter,ou=People,dc=example,dc=com
cn: Sam Carter
mail: scarter@example.com
entryLevelRights: vadn
attributeLevelRights: cn:rscwo, mail:rscwo, initials:rscwo
```

It is possible to specify a non-existent attribute in the *attributeList*, as with the ***initials*** attribute in Example 13.6, "Get Effective Rights Results for Specific Attributes", to see the rights which are available, similar to using an asterisk to list all attributes.

The Directory Manager can also list the rights for all of the attributes available to a specific object class. This option has the format *attribute@objectClass*. This returns two entries; the first for the specified GER subject and the second for a template entry for the object class.

Example 13.7. Get Effective Rights Results for an Attribute within an Object Class

```
ldapsearch -D "cn=directory manager" -w secret -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
uidNumber@posixAccount

 ... snip ...

dn: cn=template_posixaccount_objectclass,uid=scarter,ou=people,dc=example,dc=com
uidnumber: (template_attribute)
entryLevelRights: v
attributeLevelRights: uidNumber:rsc
```

NOTE

Using the search format *attribute@objectClass* is only available if the requester (**-D**) is the Directory Manager.

Using an asterisk (*****) instead of a specific attribute returns all of the attributes (present and non-existent) for the specified GER subject and the full list of attributes for the object class template.

Example 13.8. Get Effective Rights Results for All Attributes for an Object Class

```
ldapsearch -D "cn=directory manager" -w secret -b "uid=scarter,ou=people,dc=example,dc=com" -E
```

```
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" *@posixaccount

 ... snip ...

dn: cn=template_posixaccount_objectclass,uid=scarter,ou=people,dc=example,dc=com
objectClass: posixaccount
objectClass: top
homeDirectory: (template_attribute)
gidNumber: (template_attribute)
uidNumber: (template_attribute)
uid: (template_attribute)
cn: (template_attribute)
entryLevelRights: v
attributeLevelRights: cn:rsc, uid:rsc, uidNumber:rsc, gidNumber:rsc, homeDirectory:rsc, objectClass:rsc,
userPassword:none, loginShell:rsc, gecos:rsc, description:rsc, aci:rsc
```

13.7.3.4. Examples of Get Effective Rights Searches for Non-Existent Entries

An administrator may want to check what rights a specific user (**jsmith**) would have to a non-existent user, based on the existing access control rules. For checking non-existent entries, the server generates a fake entry within that subtree. For example, to check for the fake entry **cn=joe new user,cn=accounts,ou=people,dc=example,dc=com**, the server creates **cn=template,cn=accounts,ou=people,dc=example,dc=com**.

For checking a non-existent entry, the get effective rights search can use a specified object class to generate a template entry with all of the potential attributes of the (non-existent) entry. For **cn=joe new user,cn=accounts,ou=people,dc=example,dc=com** with a **person** object class (**@person**), the server generates **cn=template_person_objectclass,cn=accounts,ou=people,dc=example,dc=com**.

When the server creates the template entry, it uses the first MUST attribute in the object class definition to create the RDN attribute (or it uses MAY if there is no MUST attribute). However, this may result in an erroneous RDN value which, in turn, violates or circumvents established ACIs for the given subtree. In that case, it is possible to specify the RDN value to use by passing it with the object class. This has the form *@objectclass:rdn_attribute*.

For example, to check the rights of **scarter** for a non-existent Posix entry with ***uidNumber*** as its RDN:

```
ldapsearch -D "cn=directory manager" -w secret -b "ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)"
@posixaccount:uidnumber

dn: uidNumber=template_posixaccount_objectclass,ou=people,dc=example,dc=com
entryLevelRights: v
attributeLevelRights: description:rsc, gecos:rsc, loginShell:rsc, userPassword
 :rsc, objectClass:rsc, homeDirectory:rsc, gidNumber:rsc, uidNumber:rsc, uid:
 rsc, cn:rsc
```

13.7.3.5. Examples of Get Effective Rights Searches for Operational Attributes

Operational attributes are not returned in regular **ldapsearch**es, including get effective rights searches. To return the information for the operational attributes, use the plus sign (**+**). This returns only the operational attributes that can be used in the entry.

Example 13.9. Get Effective Rights Results for Operational Attributes

```
ldapsearch -D "cn=directory manager" -w secret -x -b "uid=scarter,ou=people,dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "+"

dn: uid=scarter,ou=People,dc=example,dc=com
entryLevelRights: vadn
attributeLevelRights: nsICQStatusText:rscwo, passwordGraceUserTime:rscwo, pwdGraceUserTime:rscwo,
nsYIMStatusText:rscwo, modifyTimestamp:rscwo, passwordExpWarned:rscwo, pwdExpirationWarned:rscwo,
entrydn:rscwo, aci:rscwo, nsSizeLimit:rscwo, nsAccountLock:rscwo, passwordExpirationTime:rscwo,
entryid:rscwo, nsSchemaCSN:rscwo, nsRole:rscwo, retryCountResetTime:rscwo, ldapSchemas:rscwo,
nsAIMStatusText:rscwo, copiedFrom:rscwo, nsICQStatusGraphic:rscwo, nsUniqueId:rscwo,
creatorsName:rscwo, passwordRetryCount:rscwo, dncomp:rscwo, nsTimeLimit:rscwo, passwordHistory:rscwo,
pwdHistory:rscwo, nscpEntryDN:rscwo, subschemaSubentry:rscwo, nsYIMStatusGraphic:rscwo,
hasSubordinates:rscwo, pwdpolicysubentry:rscwo, nsAIMStatusGraphic:rscwo, nsRoleDN:rscwo,
createTimestamp:rscwo, accountUnlockTime:rscwo, copyingFrom:rscwo, nsLookThroughLimit:rscwo,
nsds5ReplConflict:rscwo, modifiersName:rscwo, parentid:rscwo, passwordAllowChangeTime:rscwo,
nsBackendSuffix:rscwo, nsIdleTimeout:rscwo, ldapSyntaxes:rscwo, numSubordinates:rscwo
```

13.7.3.6. Examples of Get Effective Rights Results and Access Control Rules

Get effective rights are returned according to whatever ACLs are in effect for the get effective rights subject entry.

For example, this ACL is set and, for the purposes of this example, it is the only ACL set:

```
dn: dc=example,dc=com
objectClass: top
objectClass: domain
dc: example
aci: (target=ldap:///ou=Accounting,dc=example,dc=com)(targetattr="*")(version
 3.0; acl "test acl"; allow (read,search,compare) (userdn = "ldap:///anyone") ;)

dn: ou=Accounting,dc=example,dc=com
objectClass: top
objectClass: organizationalUnit
ou: Accounting
```

Because the ACL does not include the **dc=example,dc=com** subtree, the get effective rights search shows that the user does not have any rights to the **dc=example,dc=com** entry:

Example 13.10. Get Effective Rights Results with No ACL Set (Directory Manager)

```
ldapsearch -D "cn=directory manager" -w secret -b "dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "*@person"

dn: cn=template_person_objectclass,uid=scarter,ou=people,dc=example,dc=com
objectClass: person
objectClass: top
cn: (template_attribute)
sn: (template_attribute)
description: (template_attribute)
seeAlso: (template_attribute)
telephoneNumber: (template_attribute)
userPassword: (template_attribute)
entryLevelRights: none
attributeLevelRights: sn:none, cn:none, objectClass:none, description:none, seeAlso:none,
telephoneNumber:none, userPassword:none, aci:none
```

If a regular user, rather than Directory Manager, tried to run the same command, the result would simply be blank.

Example 13.11. Get Effective Rights Results with No ACL Set (Regular User)

```
$ ldapsearch -D "uid=scarter,ou=people,dc=example,dc=com" -w secret -b "dc=example,dc=com" -E
'!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=scarter,ou=people,dc=example,dc=com' "(objectclass=*)" "*@person"

$
```

13.7.4. Using Get Effective Rights from the Console

1. Open the **Directory** tab, and right-click the entry of which to check the rights.
2. Select **Advanced Properties** from the drop-down menu.
3. Check the **Show effective rights** check box.

4. Beside each attribute, the attribute-level get effective rights are displayed. The entry-level rights are shown beneath the entry's DN.

The attribute-level effective rights (**r**, **s**, **c**, **w**, **o**) appear next to the attributes. The entry-level rights (**v**, **a**, **d**, **n**) appear under the full DN for the entry in the lower left-hand corner of the **Property Editor**.

If you check the **Show all allowed attributes** check box, then the effective rights for those attributes appear next to the additional attributes, even though they do not have values.

13.7.5. Get Effective Rights Return Codes

If the criticality is not set for a get effective rights search and an error occurs, the regular entry information is returned, but, in place of rights for **entryLevelRights** and **attributeLevelRights**, an error code is returned. This code can give information on the configuration of the entry that was queried. Table 13.8, "Returned Result Codes" summarizes the error codes and the potential configuration information they can relay.

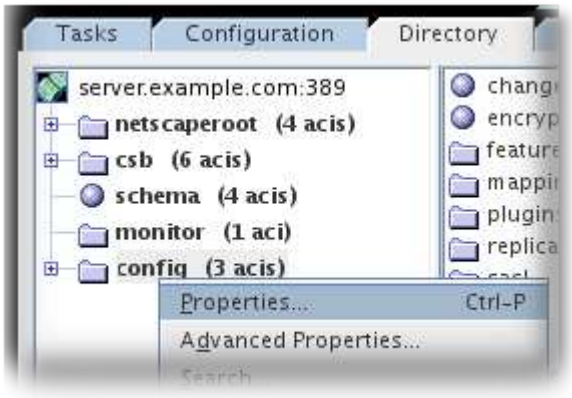
Table 13.8. Returned Result Codes

Code	Description
0	Successfully completed.
1	Operation error.
12	The critical extension is unavailable. If the criticality expression is set to **true** and effective rights do not exist on the entry being queried, then this error is returned.
16	No such attribute. If an attribute is specifically queried for access rights but that attribute does not exist in the schema, this error is returned.
17	Undefined attribute type.
21	Invalid attribute syntax.
50	Insufficient rights.
52	Unavailable.
53	Unwilling to perform.
80	Other.

13.8. LOGGING ACCESS CONTROL INFORMATION

To obtain information on access control in the error logs, you must set the appropriate log level. To set the error log level from the Console:

1. In the Console, click the **Directory** tab, right-click the config node, and choose **Properties** from the pop-up menu.

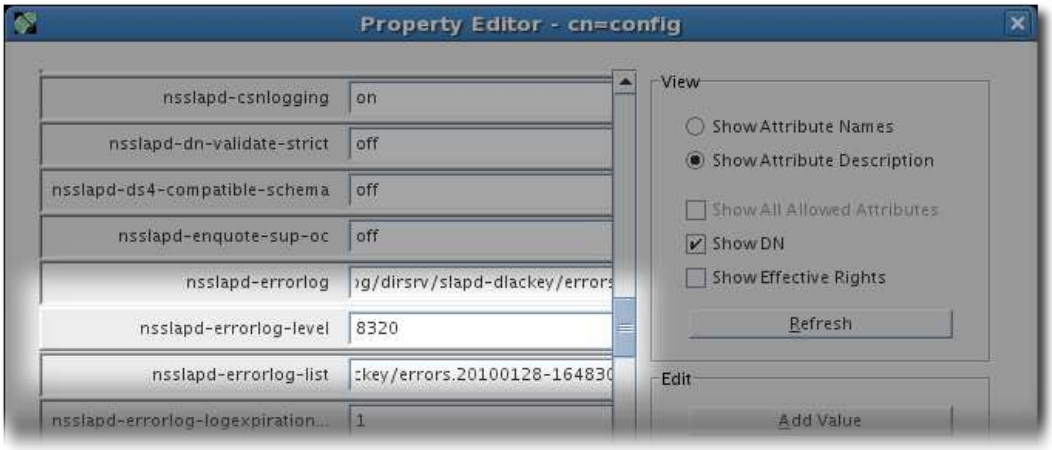


   This displays the **Property Editor** for the **cn=config** entry.

2. Scroll down the list of attribute value pairs to locate the **nsslapd-errorlog-level** attribute.

3. Add **128** to the value already displayed in the **nsslapd-errorlog-level** value field.

   For example, if the value already displayed is **8192** (replication debugging), change the value to **8320**. For complete information on error log levels, see the *Directory Server Configuration and Command-Line Tool Reference*.



4. Click **OK** to dismiss the **Property Editor**.

13.9. ACCESS CONTROL USAGE EXAMPLES

The examples provided in this section illustrate how an imaginary ISP company, Example Corp., would implement its access control policy. All the examples explain how to perform a given task from the Console and using an LDIF file.

Example Corp.'s business is to offer a web hosting service and Internet access. Part of Example Corp.'s web hosting service is to host the directories of client companies. Example Corp. actually hosts and partially manages the directories of two medium-sized companies, **HostedCompany1** and **HostedCompany2**. It also provides Internet access to a number of individual subscribers.

These are the access control rules that Example Corp. wants to put in place:

- Grant anonymous access for read, search, and compare to the entire **example** tree for Example Corp. employees (Section 13.9.1, "Granting Anonymous Access").
- Grant write access to Example Corp. employees for personal information, such as ***homePhone*** and ***homePostalAddress*** (Section 13.9.2, "Granting Write Access to Personal Entries").
- Grant Example Corp. employees the right to add any role to their entry, except certain critical roles (Section 13.9.3, "Restricting Access to Key Roles").
- Grant the **example.com Human Resources** group all rights on the entries in the **People** branch (Section 13.9.4, "Granting a Group Full Access to a Suffix").
- Grant all Example Corp. employees the right to create group entries under the **Social Committee** branch of the directory and to delete group entries that they own (Section 13.9.5, "Granting Rights to Add and Delete Group Entries").
- Grant all Example Corp. employees the right to add themselves to group entries under the **Social Committee** branch of the directory (Section 13.9.9, "Allowing Users to Add or Remove Themselves from a Group").
- Grant access to the directory administrator (role) of **HostedCompany1** and **HostedCompany2** on their respective branches of the directory tree, with certain conditions such as SSL authentication, time and date restrictions, and specified location (Section 13.9.6, "Granting Conditional Access to a Group or Role").
- Deny individual subscribers access to the billing information in their own entries (Section 13.9.7, "Denying Access").
- Grant anonymous access to the world to the individual subscribers subtree, except for subscribers who have specifically requested to be unlisted. (This part of the directory could be a consumer server outside of the firewall and be updated once a day.) See Section 13.9.1, "Granting Anonymous Access" and Section 13.9.8, "Setting a Target Using Filtering".

13.9.1. Granting Anonymous Access

Most directories are run such that you can anonymously access at least one suffix for read, search, or compare. For example, you might want to set these permissions if you are running a corporate personnel directory that you want employees to be able to search, such as a phonebook. This is the case at Example Corp. internally and is illustrated in Section 13.9.1.1, "ACI "Anonymous example.com"".

As an ISP, Example Corp. also wants to advertise the contact information of all of its subscribers by creating a public phonebook accessible to the world. This is illustrated in Section 13.9.1.2, “ACI "Anonymous World"”.

13.9.1.1. ACI "Anonymous example.com"

In LDIF, to grant read, search, and compare permissions to the entire Example Corp. tree to Example Corp. employees, write the following statement:

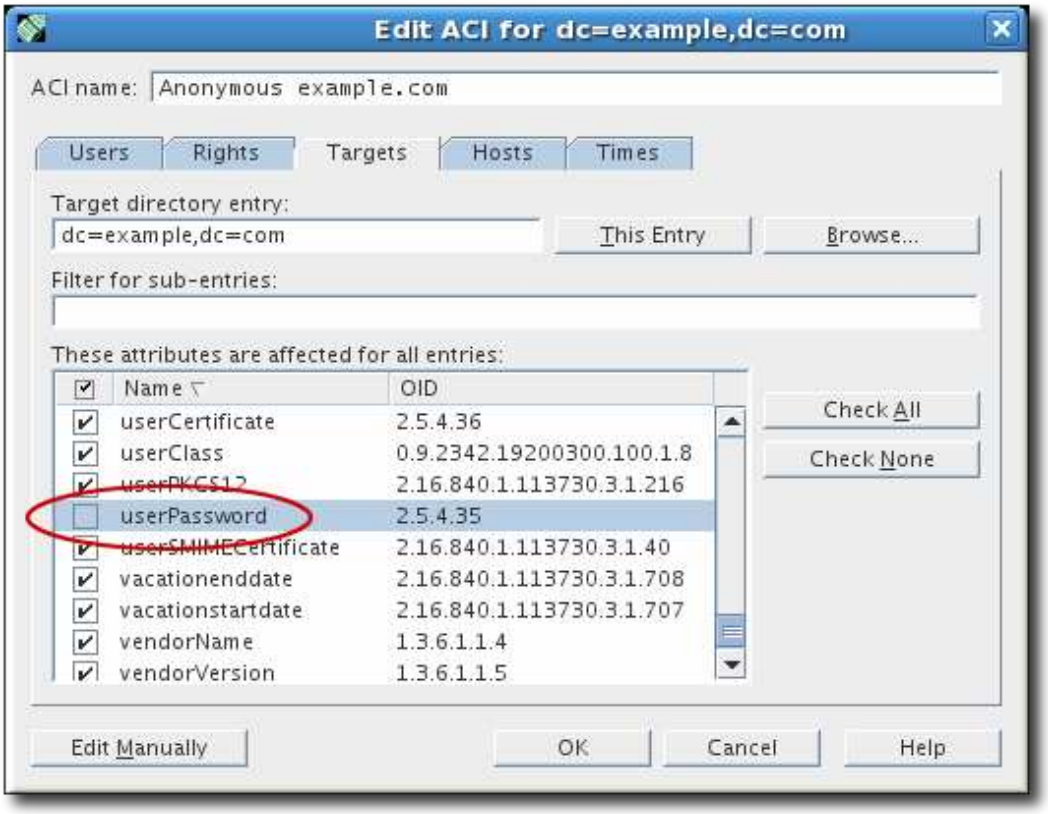
```
aci: (targetattr !="userPassword")(version 3.0; acl "Anonymous
    Example"; allow (read, search, compare) userdn= "ldap:///anyone"
    and dns="*.example.com";)
```

This example assumes that the ***aci*** attribute is added to the **dc=example,dc=com** entry. The ***userPassword*** attribute is excluded from the scope of the ACI.

From the Console:

1. In the **Directory** tab, right-click the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab in the **ACI name** field, type **Anonymous example.com**. Check that **All Users** opens in the list of users granted access permission.
4. In the **Rights** tab, select the check boxes for **read**, **compare**, and **search** rights. Make sure the other check boxes are clear.

5. In the **Targets** tab, click **This Entry** to display the **dc=example,dc=com** suffix in the **Target directory entry** field. In the attribute table, locate the ***userPassword*** attribute, and clear the corresponding check box.



All other check boxes should be selected. This task is easier if you click the **Name** header to organize the list of attributes alphabetically.

6. In the **Hosts** tab, click **Add**, and in the **DNS host filter** field, type ***.example.com**. Click **OK** to dismiss the dialog box.

7. Click **OK** in the **Access Control Editor** window.

13.9.1.2. ACI "Anonymous World"

In LDIF, to grant read and search access of the individual subscribers subtree to the world, while denying access to information on unlisted subscribers, write the following statement:

```
aci: (targetfilter= "(!(unlistedSubscriber=yes))")
    (targetattr="homePostalAddress || homePhone || mail") (version
    3.0; acl "Anonymous World"; allow (read, search) userdn="ldap:///anyone";)
```

This example assumes that the ACI is added to the **ou=subscribers,dc=example,dc=com** entry. It also assumes that every subscriber entry has an **unlistedSubscriber** attribute which is set to **yes** or **no**. The target definition filters out the unlisted subscribers based on the value of this attribute. For details on the filter definition, see Section 13.9.8, "Setting a Target Using Filtering".

From the Console:

1. In the **Directory** tab, right-click the **Subscribers** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.

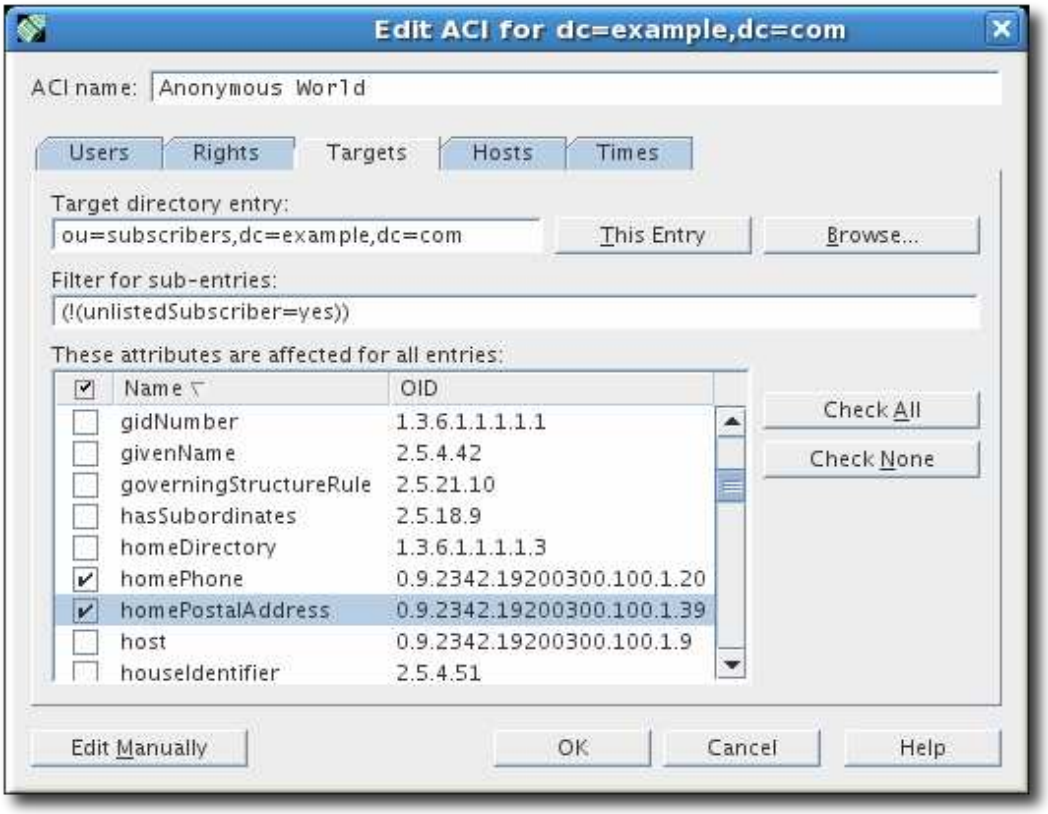
2. Click **New** to display the **Access Control Editor**.

3. In the **Users/Groups** tab, in the **ACI name** field, type **Anonymous World**. Check that **All Users** opens in the list of users granted access permission.

4. In the **Rights** tab, select the check boxes for **read** and **search** rights. Make sure the other check boxes are clear.

5. In the **Targets** tab, click **This Entry** to display the **ou=subscribers,dc=example,dc=com** suffix in the **Target directory entry** field.

6. In the **Filter for subentries** field, enter a filter which excludes unlisted subscribers (**(!(unlistedSubscriber=yes))**).



7. In the attribute table, select the check boxes for the ***homePhone***, ***homePostalAddress***, and ***mail*** attributes.

   All other check boxes should be clear; if it is easier, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

8. Click **OK**.

13.9.2. Granting Write Access to Personal Entries

Many directory administrators want to allow internal users to change some but not all of the attributes in their own entry. The directory administrators at Example Corp. want to allow users to change their own password, home telephone number, and home address, but nothing else. This is illustrated in Section 13.9.2.1, "ACI "Write example.com"".

It is also Example Corp.'s policy to let their subscribers update their own personal information in the **example** tree, provided that they establish an SSL connection to the directory. This is illustrated in Section 13.9.2.2, "ACI "Write Subscribers"".

13.9.2.1. ACI "Write example.com"

NOTE

By setting this permission, users will also have the right to delete attribute values.

Granting Example Corp. employees the right to update their password, home telephone number, and home address has the following statement in LDIF:

```
aci: (targetattr="userPassword || homePhone ||
    homePostalAddress") (version 3.0; acl "Write example.com"; allow
    (write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the **ou=people,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **people** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.

2. Click **New** to display the **Access Control Editor**.

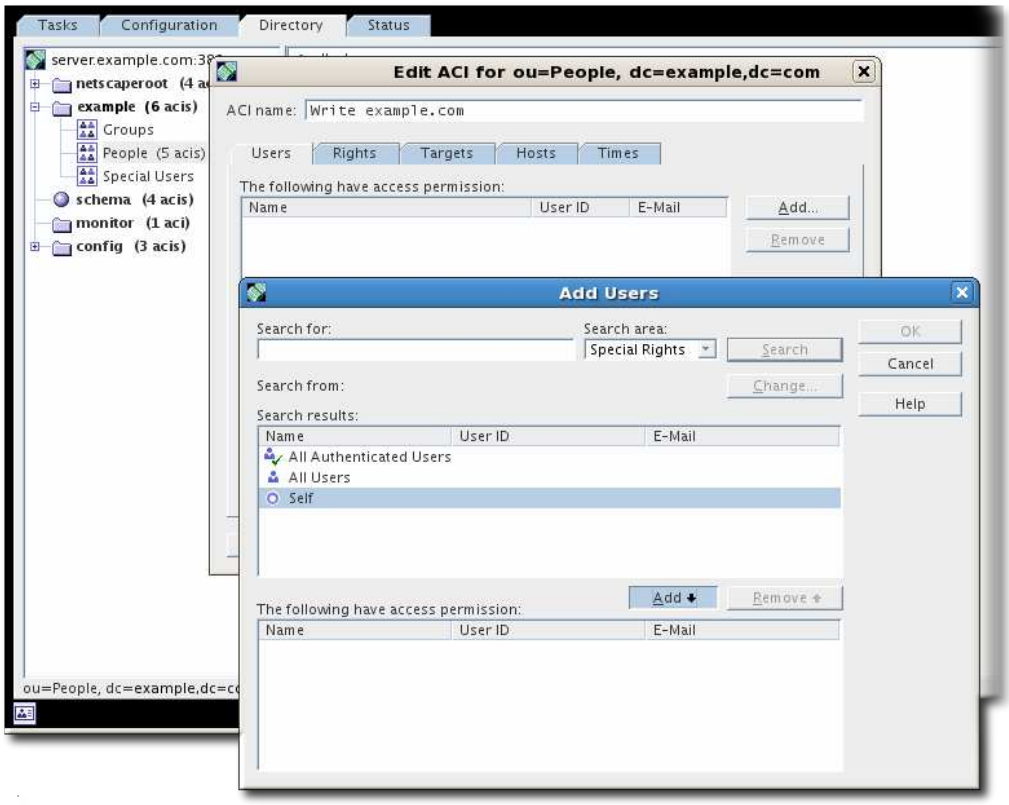
3. In the **Users/Groups** tab, in the **ACI name** field, type **Write example.com**. In the list of users granted access permission:

    1. Remove **All Users** from the targeted user list.

    2. Click **Add**.

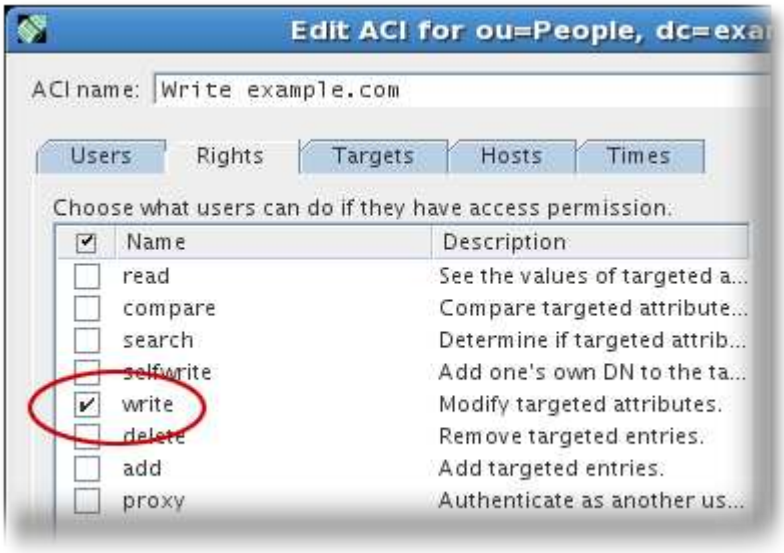
    3. Set the **Search** area to **Special Rights**, and select **Self** from the search results list.

    4. Click the **Add** button to list **Self** in the list of users who are granted access permission.

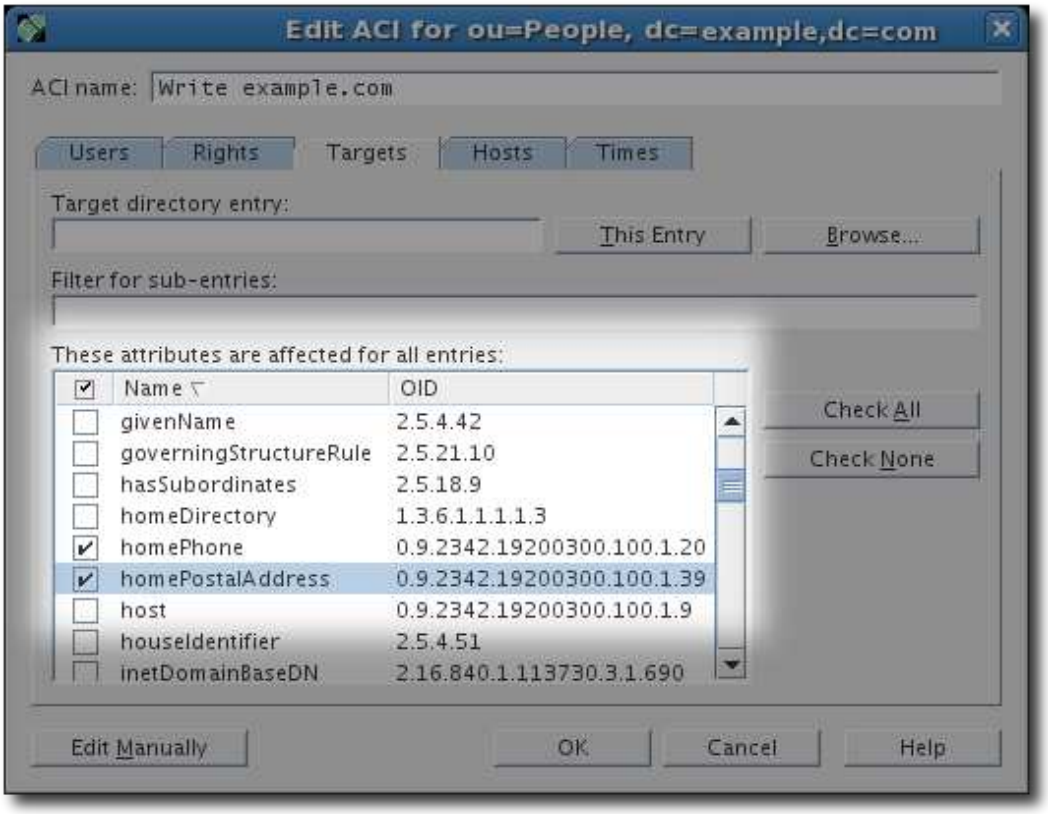


    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.

4. In the **Rights** tab, select the check box for **write** right. Make sure the other check boxes are clear.

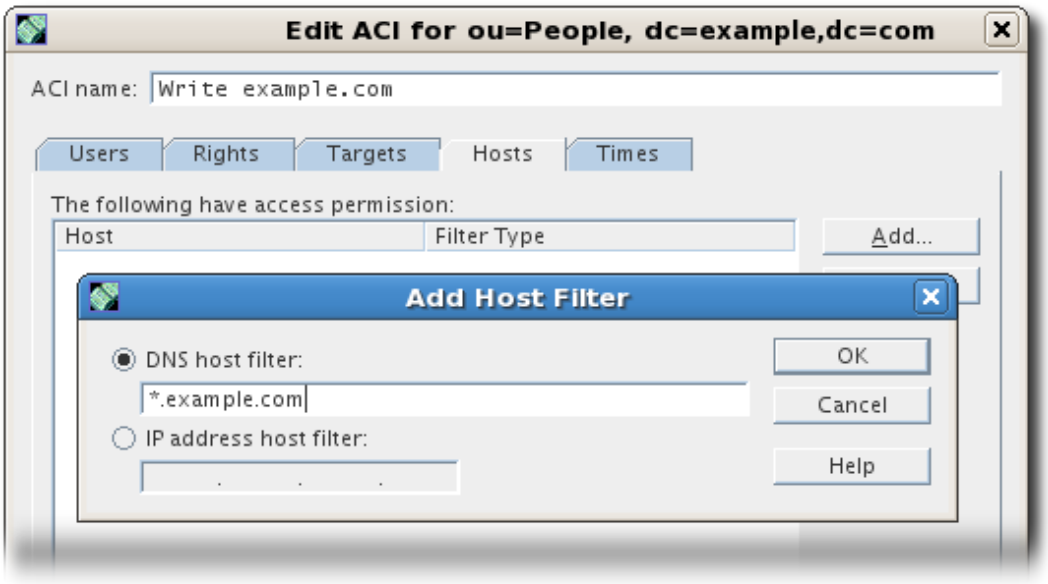


5. In the **Targets** tab, click **This Entry** to display the **ou=people,dc=example,dc=com** suffix in the **Target directory entry** field. In the attribute table, select the check boxes for the ***homePhone***, ***homePostalAddress***, and ***userPassword*** attributes.



All other check boxes should be clear; if it is easier, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type ***.example.com**. Click **OK** to dismiss the dialog box.



7. Click **OK** in the **Access Control Editor** window.

13.9.2.2. ACI "Write Subscribers"

NOTE

By setting this permission, you are also granting users the right to delete attribute values.

In LDIF, to grant Example Corp. subscribers the right to update their password and home telephone number, write the following statement:

```
aci: (targetattr = "homePhone || homePostalAddress || mail")
(target = "ldap:///ou=Subscribers,dc=example,dc=com")
(targetfilter = (!(unlistedSubscriber=yes)) )
(version 3.0;
acl "Write Subscribers";
allow (write)
(userdn = "ldap:///self") and authmethod="ssl";
;)
```

This example assumes that the **aci** is added to the **ou=subscribers,dc=example,dc=com** entry.

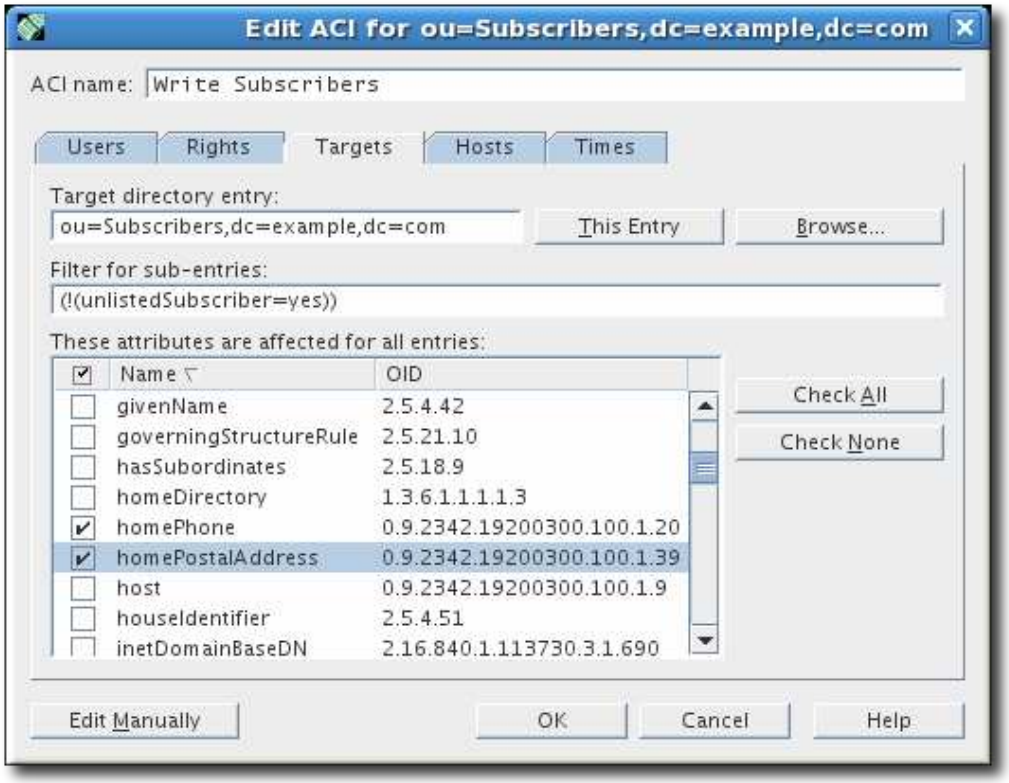
Example Corp. subscribers do not have write access to their home address because they might delete the attribute, and Example Corp. needs that information for billing. Therefore, the home address is business-critical information.

From the Console:

1. In the **Directory** tab, right-click the **subscribers** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Write Subscribers**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area to **Special Rights**, and select **Self** from the search results list.
    4. Click the **Add** button to list **Self** in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the check box for **write**. Make sure the other check boxes are clear.
5. In the **Targets** tab, click **This Entry** to display the **ou=subscribers,dc=example,dc=com** suffix in the **Target directory entry** field.
    1. In the **Filter for subentries** field, type a filter so that only listed subscribers are included:

        ```
        (!(unlistedSubscriber=yes))
        ```

    2. In the attribute table, select the check boxes for the ***homePhone***, ***homePostalAddress***, and ***mail*** attributes.



All other check boxes should be clear; if necessary, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

3. Optionally, to require users to authenticate using SSL, switch to manual editing by clicking the **Edit Manually** button, and add **authmethod=ssl** to the LDIF statement:

```
(targetattr = "homePhone || homePostalAddress || mail")
(target = "ldap:///ou=Subscribers,dc=example,dc=com")
(targetfilter = (!(unlistedSubscriber=yes)) )
(version 3.0;
acl "Write Subscribers";
allow (write)
(userdn = "ldap:///self") and authmethod="ssl";
;)
```

6. Click **OK**.

13.9.3. Restricting Access to Key Roles

You can use role definitions in the directory to identify functions that are critical to your business, the administration of your network and directory, or another purpose.

For example, you might create a **superAdmin** role by identifying a subset of your system administrators that are available at a particular time of day and day of the week at corporate sites worldwide, or you might want to create a **First Aid** role that includes all members of staff on a particular site that have done first aid training. For information on creating role definitions, see Section 6.2, “Using Roles”.

When a role gives any sort of privileged user rights over critical corporate or business functions, consider restricting access to that role. For example, at Example Corp., employees can add any role to their own entry except the **superAdmin** role. This is illustrated in Section 13.9.3.1, “ACI "Roles"”.

13.9.3.1. ACI "Roles"

In LDIF, to grant Example Corp. employees the right to add any role to their own entry except the **superAdmin** role, write the following statement:

```
aci: (targetattr = "nsroledn")
    (targetattrfilters="add=nsroledn:(nsroledn !=
    "cn=superAdmin,dc=example,dc=com")") (version 3.0; acl "Roles";
    allow (write) userdn= "ldap:///self" and dns="*.example.com";)
```

This example assumes that the ACI is added to the **ou=people,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **people** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Roles**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area in the **Add Users and Groups** dialog box to **Special Rights**, and select **Self** from the search results list.
    4. Click the **Add** button to list **Self** in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the check box for **write**. Make sure the other check boxes are clear.
5. In the **Targets** tab, click **This Entry** to use the **ou=people,dc=example,dc=com** suffix in the **Target directory entry** field.
6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type ***.example.com**. Click **OK** to dismiss the dialog box.
7. To create the value-based filter for roles, switch to manual editing by clicking the **Edit Manually** button. Add the following to the beginning of the LDIF statement:

    ```
    (targetattrfilters="add=nsroledn:(nsroledn != "cn=superAdmin,dc=example,dc=com")")
    ```

    The LDIF statement should read as follows:

    ```
    (targetattrfilters="add=nsroledn:(nsroledn != "cn=superAdmin,
        dc=example,dc=com")") (targetattr = "*") (target = "ldap:///
        ou=people,dc=example,dc=com") (version 3.0; acl "Roles";
        allow (write) (userdn = "ldap:///self") and (dns="*.example.com");)
    ```

8. Click **OK**.

13.9.4. Granting a Group Full Access to a Suffix

Most directories have a group that is used to identify certain corporate functions. These groups can be given full access to all or part of the directory. By applying the access rights to the group, you can avoid setting the access rights for each member individually. Instead, you grant users these access rights simply by adding them to the group.

For example, when the Directory Server is set up with a typical process, an administrators group with full access to the directory is created by default.

At Example Corp., the **Human Resources** group is allowed full access to the **ou=people** branch of the directory so that they can update the employee database. This is illustrated in Section 13.9.4.1, “ACI "HR"”.

13.9.4.1. ACI "HR"

In LDIF, to grant the HR group all rights on the employee branch of the directory, use the following statement:

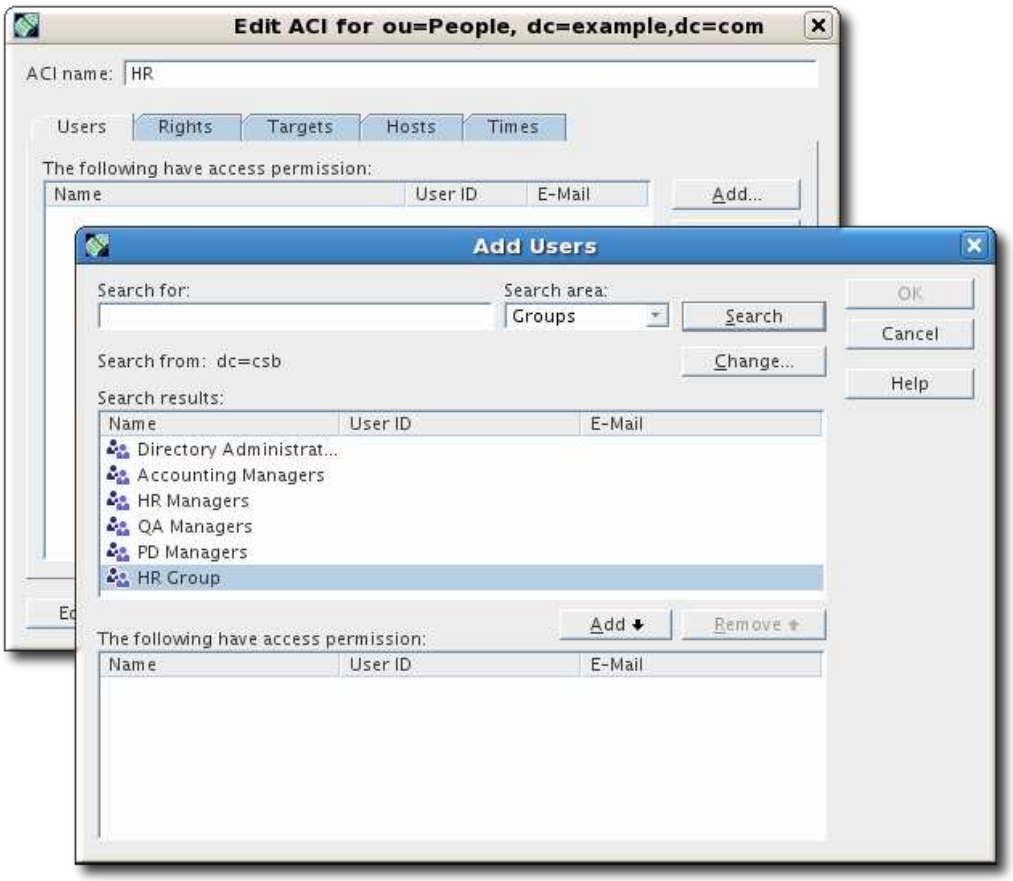
```
aci: (version 3.0; acl "HR"; allow (all) userdn=
    "ldap:///cn=HRgroup,ou=people,dc=example,dc=com";)
```

This example assumes that the ACI is added to the **ou=people,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **people** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.

2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **HR**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area to **Users and Groups**, and type **HRgroup** in the **Search for** field.



    This example assumes that you have created an HR group or role. For more information on groups and roles, see Chapter 6, *Organizing and Grouping Entries*.

    4. Click the **Add** button to list the HR group in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, click the **Check All** button.

    All check boxes are selected, except for proxy rights.

5. Click **OK**.

13.9.5. Granting Rights to Add and Delete Group Entries

Some organizations want to allow employees to create entries in the tree if it can increase their efficiency or if it can contribute to the corporate dynamics.

At Example Corp., there is an active social committee that is organized into various clubs, such as tennis, swimming, and skiing. Any Example Corp. employee can create a group entry representing a new club. This is illustrated in Section 13.9.5.1, “ACI "Create Group"” . Any Example Corp. employee can become a member of one of these groups. This is illustrated in Section 13.9.9.1, “ACI "Group Members"” under Section 13.9.9, “Allowing Users to Add or Remove Themselves from a Group”. Only the group owner can modify or delete a group entry. This is illustrated in Section 13.9.5.2, “ACI "Delete Group"” .

13.9.5.1. ACI "Create Group"

In LDIF, to grant Example Corp. employees the right to create a group entry under the **ou=Social Committee** branch, write the following statement:

```
aci: (target="ldap:///ou=social committee,dc=example,dc=com)
   (targetattrfilters="add=objectClass:(objectClass=groupOfNames)")
   (version 3.0; acl "Create Group"; allow (add)
   (userdn= "ldap:///uid=*,ou=people,dc=example,dc=com")
   and dns="*.example.com";)
```

NOTE

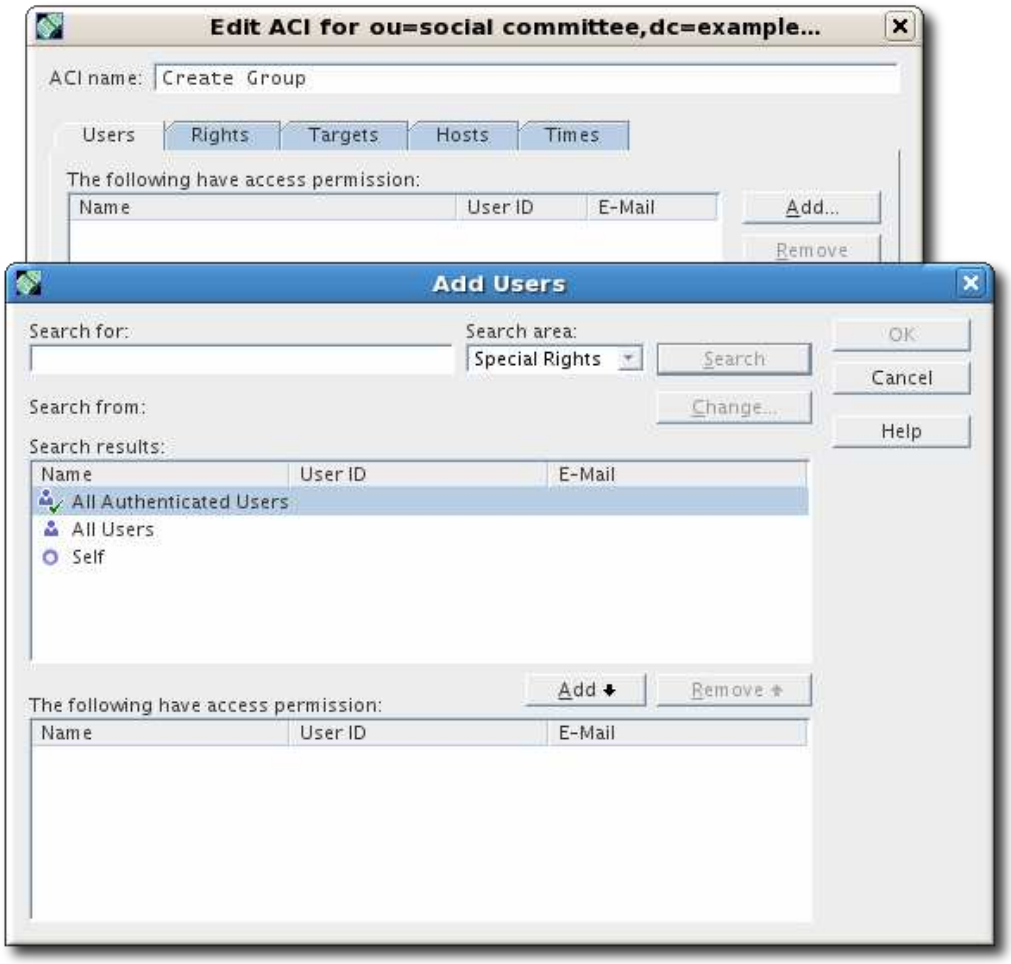
This ACI does not grant write permission, which means that the entry creator cannot modify the entry.

This example assumes that the ACI is added to the **ou=social committee,dc=example,dc=com** entry.

From the Console:

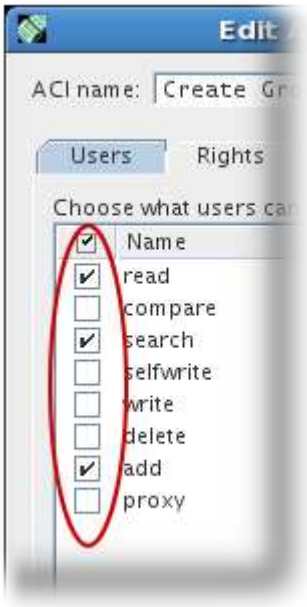
1. In the **Directory** tab, right-click the **Social Committee** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Create Group**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area to **Special Rights**, and select **All Authenticated Users** from the search results list.

4. Click the **Add** button to list **All Authenticated Users** in the list of users who are granted access permission.



5. Click **OK** to dismiss the **Add Users and Groups** dialog box.

4. In the **Rights** tab, select the check boxes for add, search, and read. Make sure the other check boxes are clear.



5. In the **Targets** tab, click **This Entry** to display the **ou=social committee,dc=example,dc=com** suffix in the **Target directory entry** field.

6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **DNS host filter** field, type ***.example.com**. Click **OK** to dismiss the dialog box.

7. To create the value-based filter that allows employees to add only group entries to this subtree, click the **Edit Manually** button. Add the following to the beginning of the LDIF statement:

   ```
   (targetattrfilters="add=objectClass:(objectClass=groupOfNames)")
   ```

   The LDIF statement should read as follows:

   ```
   (targetattrfilters="add=objectClass:(objectClass=groupOfNames)")
       (targetattr = "*") (target="ldap:///ou=social committee,dc=example,dc=com)
       (version 3.0; acl "Create Group"; allow (read,search,add)
       (userdn= "ldap:///all") and (dns="*.example.com"); )
   ```

8. Click **OK**.

13.9.5.2. ACI "Delete Group"

In LDIF, to grant Example Corp. employees the right to modify or delete a group entry which they own under the **ou=Social Committee** branch, write the following statement:

```
aci: (target="ou=social committee,dc=example,dc=com)
    (targetattrfilters="del=objectClass:(objectClass=groupOfNames)")
    (version 3.0; acl "Delete Group"; allow (delete) userattr=
    "owner#GROUPDN";)
```

This example assumes that the **aci** is added to the **ou=social committee,dc=example,dc=com** entry.

NOTE

Using the Console is not an effective way of creating this ACI because it requires manually editing the ACI to create the target filter and to check group ownership.

13.9.6. Granting Conditional Access to a Group or Role

In many cases, when you grant a group or role privileged access to the directory, you want to ensure that those privileges are protected from intruders trying to impersonate your privileged users. Therefore, in many cases, access control rules that grant critical access to a group or role are often associated with a number of conditions.

Example Corp. has created a directory administrator role for each of its hosted companies, **HostedCompany1** and **HostedCompany2**. It wants these companies to be able to manage their own data and implement their own access control rules while securing it against intruders. For this reason, **HostedCompany1** and **HostedCompany2** have full rights on their respective branches of the directory tree, provided the following conditions are fulfilled:

- Connection authenticated using SSL
- Access requested between 8 a.m. and 6 p.m., Monday through Thursday
- Access requested from a specified IP address for each company

These conditions are illustrated in a single ACI for each company, **HostedCompany1** and **HostedCompany2**. Because the content of these ACIs is the same, the examples below illustrate the **HostedCompany1** ACI only.

13.9.6.1. ACI "HostedCompany1"

In LDIF, to grant **HostedCompany1** full access to their own branch of the directory under the requisite conditions, write the following statement:

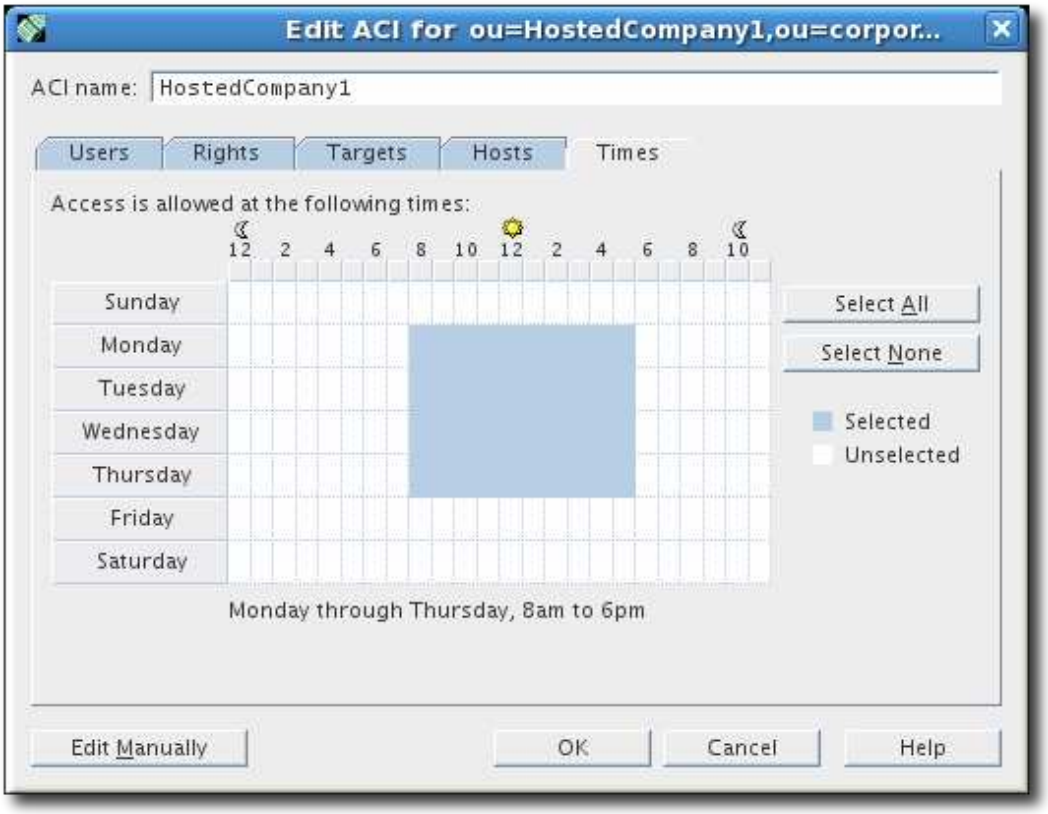
```
aci:(target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com")
    (targetattr= "*") (version 3.0; acl "HostedCompany1";allow (all)
    (roledn="ldap:///cn=DirectoryAdmin,ou=HostedCompany1,
    ou=corporate-clients,dc=example,dc=com") and
    (authmethod="ssl") and (dayofweek="Mon,Tues,Wed,Thu") and (timeofday >= "0800" and
    timeofday <= "1800") and (ip="255.255.123.234"); )
```

This example assumes that the ACI is added to the **ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **HostedCompany1** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, type **HostedCompany1** in the **ACI name** field. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area to **Users and Groups**, and type **DirectoryAdmin** in the **Search For** field.

       (This assumes that there is an administrator's role with a **cn** of **DirectoryAdmin**.)
    4. Click the **Add** button to list the administrator's role in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, click the **Check All** button.
5. In the **Targets** tab, click **This Entry** to display the **ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com** suffix in the **Target directory entry** field.
6. In the **Hosts** tab, click **Add** to display the **Add Host Filter** dialog box. In the **IP address host filter** field, type **255.255.123.234**. Click **OK**.

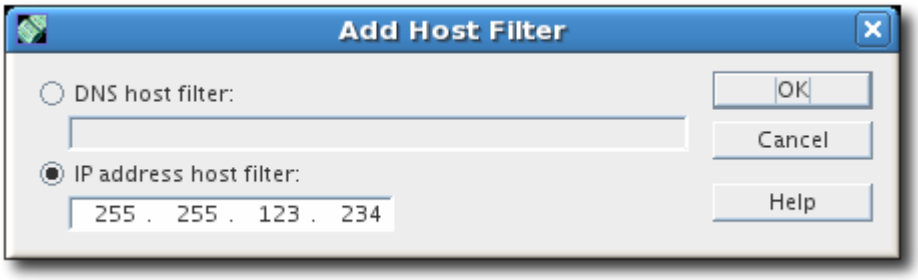


The IP address must be a valid IP address for the host machine that the **HostedCompany1** administrators use to connect to the **example** directory.

NOTE

Directory Server supports both IPv4 and IPv6 IP addresses.

7. In the **Times** tab, select the block time corresponding to Monday through Thursday and 8 a.m. to 6 p.m.



A message appears below the table that specifies the selected time block.

8. To enforce SSL authentication from **HostedCompany1** administrators, switch to manual editing by clicking the **Edit Manually** button. Add the following to the end of the LDIF statement:

```
and (authmethod="ssl")
```

The LDIF statement should be similar to the following:

```
(targetattr = "*") (target="ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com")
    (version 3.0; acl "HostedCompany1"; allow (all) (roledn=
    "ldap:///cn=DirectoryAdmin,ou=HostedCompany1,ou=corporate-clients,dc=example,dc=com") and
    (dayofweek="Mon,Tues,Wed,Thu") and (timeofday >= "0800" and timeofday <= "1800") and
    (ip="255.255.123.234") and (authmethod="ssl"); )
```

9. Click **OK**.

13.9.7. Denying Access

If your directory holds business-critical information, it may be necessary to specifically deny access to it.

For example, Example Corp. wants all subscribers to be able to read billing information such as connection time or account balance under their own entries but explicitly wants to deny write access to that information. This is illustrated in Section 13.9.7.1, “ACI "Billing Info Read"” and Section 13.9.7.2, “ACI "Billing Info Deny"”, respectively.

13.9.7.1. ACI "Billing Info Read"

In LDIF, to grant subscribers permission to read billing information in their own entry, write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
    3.0; acl "Billing Info Read"; allow (search,read) userdn=
    "ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the **ou=subscribers,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **Subscribers** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Billing Info Read**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area in the **Add Users and Groups** dialog box to **Special Rights**, and select **Self** from the search results list.
    4. Click the **Add** button to list **Self** in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.

4. In the **Rights** tab, select the check boxes for **search** and **read** rights. Make sure the other check boxes are clear.

5. In the **Targets** tab, click **This Entry** to display the **ou=subscribers,dc=example,dc=com** suffix in the **Target directory entry** field. In the attribute table, select the check boxes for the ***connectionTime*** and ***accountBalance*** attributes. (These are custom schema that Example Corp. uses for ISP account management.)

All other check boxes should be clear; if it is easier, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

This example assumes that you have added the ***connectionTime*** and ***accountBalance*** attributes to the schema.

6. Click **OK**.

13.9.7.2. ACI "Billing Info Deny"

In LDIF, to deny subscribers permission to modify billing information in their own entry, write the following statement:

```
aci: (targetattr="connectionTime || accountBalance") (version
    3.0; acl "Billing Info Deny"; deny (write) userdn="ldap:///self";)
```

This example assumes that the relevant attributes have been created in the schema and that the ACI is added to the **ou=subscribers,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **Subscribers** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Billing Info Deny**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area in the **Add Users and Groups** dialog box to **Special Rights**, and select **Self** from the search results list.
    4. Click the **Add** button to list **Self** in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the check box for **write**. Make sure the other check boxes are clear.
5. Click the **Edit Manually** button, and, in the LDIF statement that opens, change the word **allow** to **deny**.

6. In the **Targets** tab, click **This Entry** to display the **ou=subscribers,dc=example,dc=com** suffix in the **Target directory entry** field. In the attribute table, select the check boxes for the ***connectionTime*** and ***accountBalance*** attributes.

    All other check boxes should be clear; if it is easier, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

    This example assumes that the ***connectionTime*** and ***accountBalance*** attributes were added to the schema.
7. Click **OK**.

13.9.8. Setting a Target Using Filtering

To set access controls that allow access to a number of entries that are spread across the directory, consider using a filter to set the target.

NOTE

Because search filters do not directly name the object for which you are managing access, it is easy to allow or deny access to the wrong objects unintentionally, especially as your directory becomes more complex. Additionally, filters can make it difficult to troubleshoot access control problems within your directory.

For example, the following ACI grants user **bjensen** write access to the department number, home phone number, home postal address, and manager attributes for all members of the accounting organization.

```
aci: (targetattr="departmentNumber || homePhone || homePostalAddress || manager")
  (targetfilter="(uid=bjensen)") (version 3.0; acl "Filtered ACL"; allow (write)
  userdn ="ldap:///cn=*,ou=accounting,dc=example,dc=com";)
```

Before you can set these permissions, you must create the accounting branch point **ou=accounting,dc=example,dc=com**). You can create organizational unit branch points in the **Directory** tab on the Directory Server Console.

13.9.9. Allowing Users to Add or Remove Themselves from a Group

Many directories set ACIs that allow users to add or remove themselves from groups. This is useful, for example, for allowing users to add and remove themselves from mailing lists.

At Example Corp., employees can add themselves to any group entry under the **ou=social committee** subtree. This is illustrated in Section 13.9.9.1, “ACI "Group Members"”.

13.9.9.1. ACI "Group Members"

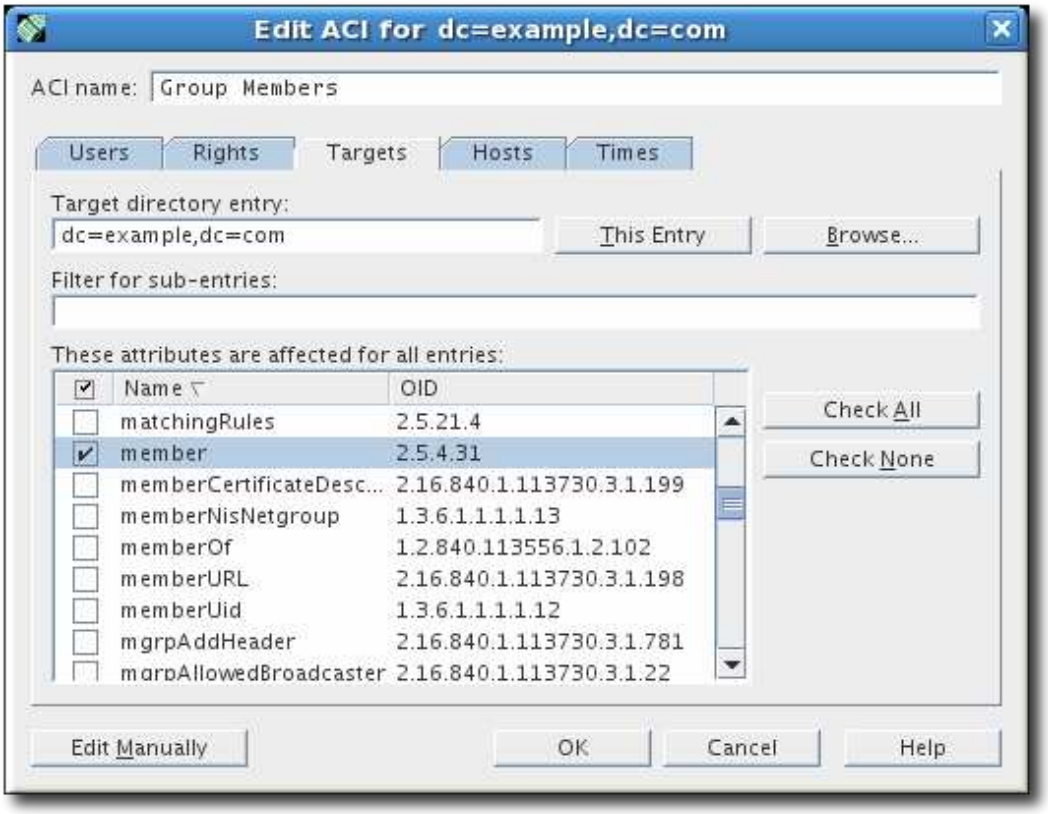
In LDIF, to grant Example Corp. employees the right to add or delete themselves from a group, write the following statement:

```
aci: (targettattr="member")(version 3.0; acl "Group Members"; allow (selfwrite)
  (userdn= "ldap:///uid=*,ou=people,dc=example,dc=com") ;)
```

This example assumes that the ACI is added to the **ou=social committee,dc=example,dc=com** entry.

From the Console:

1. In the **Directory** tab, right-click the **people** entry under the **example** node in the left navigation tree, and choose **Set Access Permissions** from the pop-up menu to display the **Access Control Manager**.
2. Click **New** to display the **Access Control Editor**.
3. In the **Users/Groups** tab, in the **ACI name** field, type **Group Members**. In the list of users granted access permission:
    1. Remove **All Users** from the targeted user list.
    2. Click **Add**.
    3. Set the **Search** area in the **Add Users and Groups** dialog box to **Special Rights**, and select **All Authenticated Users** from the search results list.
    4. Click the **Add** button to list **All Authenticated Users** in the list of users who are granted access permission.
    5. Click **OK** to dismiss the **Add Users and Groups** dialog box.
4. In the **Rights** tab, select the check box for **selfwrite**. Make sure the other check boxes are clear.
5. In the **Targets** tab, type **dc=example,dc=com** suffix in the **Target directory entry** field. In the attribute table, select the check box for the ***member*** attribute.



All other check boxes should be clear; if it is easier, click the **Check None** button to clear the check boxes for all attributes in the table, then click the **Name** header to organize them alphabetically, and select the appropriate ones.

6. Click **OK**.

13.9.10. Setting an ACI to Require a Certain Security Strength Factor for Some Operations

As mentioned in Section 13.4.8, "Requiring a Certain Level of Security in Connections" , the **ssf** keyword is used to require a secure connection and at a specific level of security. This can be a way to make changes to sensitive information only over a secure connection, like requiring that password changes be made over TLS or SASL:

```
aci: (targetattr="userPassword")(version 3.0; acl "Require secure password changes"; allow (write)
userdn="ldap:///self" and ssf>="56";)
```

13.9.11. Defining Permissions for DNs That Contain a Comma

DNs that contain commas require special treatment within your LDIF ACI statements. In the target and bind rule portions of the ACI statement, commas must be escaped by a single backslash (\). For example:

```
dn: dc=example.com Bolivia\, S.A.,dc=com
objectClass: top
objectClass: organization
aci: (target="ldap:///dc=example.com Bolivia\,S.A.,dc=com")(targetattr=*)
    (version 3.0; acl "aci 2"; allow (all)
    groupdn = "ldap:///cn=Directory Administrators,dc=example.com Bolivia\, S.A.,dc=com";)
```

13.9.12. Proxied Authorization ACI Example

Proxied authorization allows one user to bind and perform operation as another user. For example, Example Corp. has an accounting program which must be able to bind to the directory as an accounting administrator in order to write data. This authorization assumes three things:

- The client application's bind DN is **"uid=MoneyWizAcctSoftware,ou=Applications,dc=example,dc=com"**.
- The targeted subtree to which the client application is requesting access is **ou=Accounting,dc=example,dc=com**.

- An accounting administrator with access permissions to the **ou=Accounting,dc=example,dc=com** subtree exists in the directory.

In order for the client application to gain access to the accounting subtree, using the same access permissions as the accounting administrator, two ACIs must be set:

- The accounting administrator must have access permissions to the **ou=Accounting,dc=example,dc=com** subtree, so the following ACI grants all rights to the accounting administrator entry:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com") (targetattr="*")
    (version 3.0; acl "allowAll-AcctAdmin"; allow (all)
    userdn="ldap://uid=AcctAdministrator,ou=Administrators,dc=example,dc=com")
```

- There must be an ACI granting proxy rights to the client application in the directory:

```
aci: (target="ldap:///ou=Accounting,dc=example,dc=com") (targetattr="*")
    (version 3.0; acl "allow proxy-accounting software"; allow (proxy)
    userdn="ldap://uid=MoneyWizAcctSoftware,ou=Applications,dc=example,dc=com")
```

With this ACI in place, the **MoneyWizAcctSoftware** client application can bind to the directory and send an LDAP command such as **ldapsearch** or **ldapmodify** that requires the access rights of the proxy DN.

If the client performs an **ldapsearch** command, the command must include the **-e authzid=dn:** control to give the proxy account for a simple bind, as in the example, or use the **-X** *auhzid* argument for proxy authentication with a SASL bind.

```
ldapsearch -x -D "uid=MoneyWizAcctSoftware,ou=Applications,dc=example,dc=com" -w secret -e
authz="dn:uid=AcctAdministrator,ou=Administrators,dc=example,dc=com" ...
```

NOTE

Both the real user and the proxy user are recorded in the access logs. The real user is recorded as the **dn**, and the proxied user is recored as the authorization ID, **authzid**.

```
[01/Jul/2017:16:11:47 -0400] conn=1 op=0 BIND dn="uid=jsmith,ou=people,dc=example,dc=com"
authzid="uid=admin,ou=corp-accounts,dc=example,dc=com"...
```

The client or application (**MoneyWizAcctSoftware**) binds as itself but is granted the privileges of the proxy entry (**AcctAdministrator**). The client does not need the password of the proxy entry.

NOTE

There are some restrictions on binding with proxy authorization. You cannot use the Directory Manager's DN (root DN) as a proxy DN.

Additionally, if Directory Server receives more than one proxied authentication control, an error is returned to the client application, and the bind attempt is unsuccessful.

13.10. ADVANCED ACCESS CONTROL: USING MACRO ACIS

In organizations that use repeating directory tree structures, it is possible to optimize the number of ACIs used in the directory by using macros. Reducing the number of ACIs in your directory tree makes it easier to manage your access control policy and improves the efficiency of ACI memory usage.

Macros are placeholders that are used to represent a DN, or a portion of a DN, in an ACI. You can use a macro to represent a DN in the target portion of the ACI or in the bind rule portion, or both. In practice, when Directory Server gets an incoming LDAP operation, the ACI macros are matched against the resource targeted by the LDAP operation. If there is a match, the macro is replaced by the value of the DN of the targeted resource. Directory Server then evaluates the ACI normally.

13.10.1. Macro ACI Example

Figure 13.3, "Example Directory Tree for Macro ACIs" shows a directory tree which uses macro ACIs to effectively reduce the overall number of ACIs. This illustration uses repeating pattern of subdomains with the same tree structure (**ou=groups**, **ou=people**). This pattern is also repeated across the tree because the Example Corp. directory tree stores the suffixes **dc=hostedCompany2,dc=example,dc=com** and **dc=hostedCompany3,dc=example,dc=com**.

The ACIs that apply in the directory tree also have a repeating pattern. For example, the following ACI is located on the **dc=hostedCompany1,dc=example,dc=com** node:

```
aci: (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com";)
```

This ACI grants read and search rights to the **DomainAdmins** group to any entry in the **dc=hostedCompany1,dc=example,dc=com** tree.

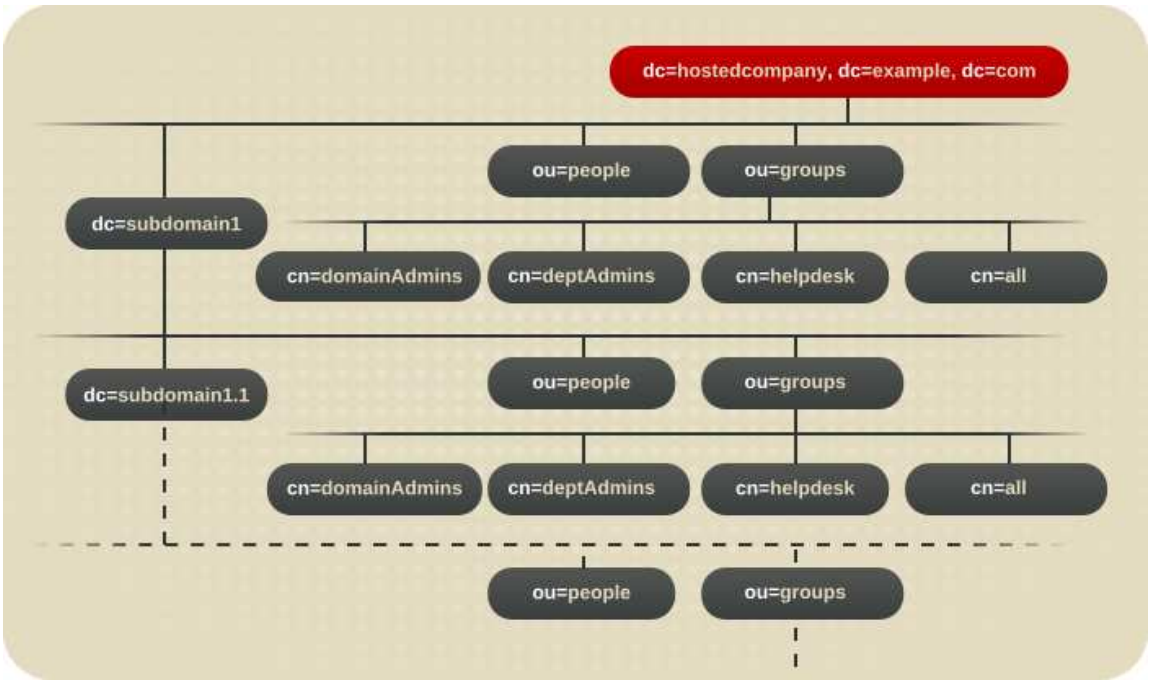


Figure 13.3. Example Directory Tree for Macro ACIs

The following ACI is located on the **dc=hostedCompany1,dc=example,dc=com** node:

```
aci: (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
   (version 3.0; acl "Domain access"; allow (read,search)
   groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com";)
```

The following ACI is located on the **dc=subdomain1,dc=hostedCompany1,dc=example,dc=com** node:

```
aci: (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com";)
```

The following ACI is located on the **dc=hostedCompany2,dc=example,dc=com** node:

```
aci: (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany2,dc=example,dc=com";)
```

The following ACI is located on the **dc=subdomain1,dc=hostedCompany2,dc=example,dc=com** node:

```
aci: (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,dc=hostedCompany2,dc=example,dc=com";)
```

In the four ACIs shown above, the only differentiator is the DN specified in the **groupdn** keyword. By using a macro for the DN, it is possible to replace these ACIs by a single ACI at the root of the tree, on the **dc=example,dc=com** node. This ACI reads as follows:

```
aci: (target="ldap:///ou=Groups,($dn),dc=example,dc=com")
    (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=com";)
```

The **target** keyword, which was not previously used, is utilized in the new ACI.

In this example, the number of ACIs is reduced from four to one. The real benefit is a factor of how many repeating patterns you have down and across your directory tree.

13.10.2. Macro ACI Syntax

Macro ACIs include the following types of expressions to replace a DN or part of a DN:

- ($dn)
- [$dn]
- ($attr.*attrName*), where *attrName* represents an attribute contained in the target entry

In this section, the ACI keywords used to provide bind credentials, such as **userdn**, **roledn**, **groupdn**, and **userattr**, are collectively called the *subject*, as opposed to the *target*, of the ACI. Macro ACIs can be used in the target part or the subject part of an ACI.

Table 13.9, "Macros in ACI Keywords" shows in what parts of the ACI you can use DN macros:

Table 13.9. Macros in ACI Keywords

Macro	ACI Keyword
($dn)	target, targetfilter, userdn, roledn, groupdn, userattr
[$dn]	targetfilter, userdn, roledn, groupdn, userattr
($attr.*attrName*)	userdn, roledn, groupdn, userattr

The following restrictions apply:

- If you use **($dn)** in **targetfilter**, **userdn**, **roledn**, **groupdn**, **userattr**, you *must* define a target that contains **($dn)**.
- If you use **[$dn]** in **targetfilter**, **userdn**, **roledn**, **groupdn**, **userattr**, you *must* define a target that contains **($dn)**.

NOTE

When using any macro, you *always* need a target definition that contains the **($dn)** macro.

You can combine the **($dn)** macro and the **($attr.***attrName***)** macro.

13.10.2.1. Macro Matching for ($dn)

The **($dn)** macro is replaced by the matching part of the resource targeted in an LDAP request. For example, you have an LDAP request targeted at the **cn=all,ou=groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com** entry and an ACI that defines the target as follows:

```
(target="ldap:///ou=Groups,($dn),dc=example,dc=com")
```

The **($dn)** macro matches with **dc=subdomain1,dc=hostedCompany1**.

When the subject of the ACI also uses **($dn)**, the substring that matches the target is used to expand the subject. For example:

```
aci: (target="ldap:///ou=*,($dn),dc=example,dc=com")
    (targetattr = "*") (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,($dn),dc=example,dc=com";)
```

In this case, if the string matching **($dn)** in the target is **dc=subdomain1,dc=hostedCompany1**, then the same string is used in the subject. The ACI is then expanded as follows:

```
aci: (target="ldap:///ou=Groups,dc=subdomain1,dc=hostedCompany1,
```

```
    dc=example,dc=com") (targetattr = "*") (version 3.0; acl "Domain
    access"; allow (read,search) groupdn="ldap:///cn=DomainAdmins,ou=Groups,
    dc=subdomain1,dc=hostedCompany1,dc=example,dc=com";)
```

Once the macro has been expanded, Directory Server evaluates the ACI following the normal process to determine whether access is granted.

13.10.2.2. Macro Matching for [$dn]

The matching mechanism for **[$dn]** is slightly different than for **($dn)**. The DN of the targeted resource is examined several times, each time dropping the left-most RDN component, until a match is found.

For example, you have an LDAP request targeted at the **cn=all,ou=groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com** subtree and the following ACI:

```
aci: (target="ldap:///ou=Groups,($dn),dc=example,dc=com")
    (targetattr = "*") (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=com";)
```

The steps for expanding this ACI are as follows:

1. **($dn)** in the target matches **dc=subdomain1,dc=hostedCompany1**.
2. **[$dn]** in the subject is replaces with **dc=subdomain1,dc=hostedCompany1**.

   The result is **groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=subdomain1,dc=hostedCompany1,dc=example,dc=com"**. If the bind DN is a member of that group, the matching process stops, and the ACI is evaluated. If it does not match, the process continues.
3. **[$dn]** in the subject is replaced with **dc=hostedCompany1**.

   The result is **groupdn="ldap:///cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com"**. In this case, if the bind DN is not a member of that group, the ACI is not evaluated. If it is a member, the ACI is evaluated.

The advantage of the **[$dn]** macro is that it provides a flexible way of granting access to domain-level administrators to *all* the subdomains in the directory tree. Therefore, it is useful for expressing a hierarchical relationship between domains.

For example, consider the following ACI:

```
aci: (target="ldap:///ou=*, ($dn),dc=example,dc=com")
    (targetattr="*")(targetfilter=(objectClass=nsManagedDomain))
    (version 3.0; acl "Domain access"; allow (read,search)
    groupdn="ldap:///cn=DomainAdmins,ou=Groups,[$dn],dc=example,dc=com";)
```

It grants access to the members of **cn=DomainAdmins,ou=Groups,dc=hostedCompany1,dc=example,dc=com** to all of the subdomains under **dc=hostedCompany1**, so an administrator belonging to that group could access a subtree like **ou=people,dc=subdomain1.1,dc=subdomain1**.

However, at the same time, members of **cn=DomainAdmins,ou=Groups,dc=subdomain1.1** would be denied access to the **ou=people,dc=hostedCompany1** and **ou=people,dc=hostedCompany1** nodes.

13.10.2.3. Macro Matching for ($attr.attrName)

The **($attr.***attrName***)** macro is always used in the subject part of a DN. For example, define the following **roledn**:

```
roledn = "ldap:///cn=DomainAdmins,($attr.ou)"
```

Now, assume the server receives an LDAP operation targeted at the following entry:

```
dn: cn=Jane Doe,ou=People,dc=HostedCompany1,dc=example,dc=com
cn: Jane Doe
sn: Doe
ou: Engineering,dc=HostedCompany1,dc=example,dc=com
...
```

In order to evaluate the **roledn** part of the ACI, the server looks at the ***ou*** attribute stored in the targeted entry and uses the value of this attribute to expand the macro. Therefore, in the example, the **roledn** is expanded as follows:

```
roledn = "ldap:///cn=DomainAdmins,ou=Engineering,dc=HostedCompany1,dc=example,dc=com"
```

The Directory Server then evaluates the ACI according to the normal ACI evaluation algorithm.

When an attribute is multi-valued, each value is used to expand the macro, and the first one that provides a successful match is used. For example:

```
dn: cn=Jane Doe,ou=People,dc=HostedCompany1,dc=example,dc=com
cn: Jane Doe
sn: Doe
ou: Engineering,dc=HostedCompany1,dc=example,dc=com
ou: People,dc=HostedCompany1,dc=example,dc=com...
```

In this case, when the Directory Server evaluates the ACI, it performs a logical OR on the following expanded expressions:

```
roledn = "ldap:///cn=DomainAdmins,ou=Engineering,dc=HostedCompany1,dc=example,dc=com"

roledn = "ldap:///cn=DomainAdmins,ou=People,dc=HostedCompany1,dc=example,dc=com"
```

13.11. SETTING ACCESS CONTROLS ON DIRECTORY MANAGER

Having an unconstrained administrative user makes sense from a maintenance perspective. The Directory Manager requires a high level of access in order to perform maintenance tasks and to response to incidents.

However, because of the power of the Directory Manager user, a certain level of access control may be advisable to prevent unauthorized access or attacks from being performed as the root user.

Regular access control rules are applied to the directory tree, the Directory Manager is not a regular user entry, so no (regular) ACIs can be applied to the Directory Manager user. ACIs are applied through a special plug-in configuration entry.

13.11.1. About Access Controls on the Directory Manager Account

Normal access control rules do not apply to the Directory Manager user. The Directory Manager is defined in the **dse.ldif** file, not in the regular user database, and so ACI targets (Section 13.3.2, “Defining Targets”) which are based on an entry within a subtree do not include the Directory Manager.

Access controls for Directory Manager are implemented through the *RootDN Access Control Plug-in* . This plug-in applies to the Directory Server configuration, and therefore can apply some access control rules to the Directory Manager entry.

The plug-in does not define a standard ACL. Some information is already implied, including the target (the Directory Manager entry) and the allowed rights (all of them). The purpose of the RootDN Access Control Plug-in is not to restrict *what* the Directory Manager can do; the purpose is to provide a level of security by limiting who can log in as Directory Manager (even with valid credentials) based on their location or time.

For this reason, the ACI for the Directory Manager only sets bind rules:

- Time-based access controls for time ranges, such as 8a.m. to 5p.m. (0800 to 1700), and day-of-week access controls, so access is only allowed on explicitly defined days. This is analogous to Section 13.4.9, “Defining Access at a Specific Time of Day or Day of Week”.
- IP address rules, where only specified IP addresses, domains, or subnets are explicitly allowed or denied. This is analogous to Section 13.4.6, “Defining Access from a Specific IP Address” .
- Host access rules, where only specified host names, domain names, or subdomains are explicitly allowed or denied. This is analogous to Section 13.4.7, “Defining Access from a Specific Domain” .

As with other access control rules, deny rules supercede allow rules.

IMPORTANT

Make sure that the Directory Manager always has the approproate level of access allowed. The Directory Manager may need to perform maintenance operations in off-hours (when user load is light) or to respond to failures. In that case, setting stringent time or day-based access control rules could prevent the Directory Manager from being able to adequately manage the directory.

13.11.2. Configuring the RootDN Access Control Plug-in

Root DN access control rules are disabled by default. The RootDN Access Control Plug-in must be enabled, and then the appropriate access control rules can be set.

NOTE

There is only *one* access control rule set for the Directory Manager, in the plug-in entry, and it applies to all access to the entire directory.

1. Enable the RootDN Access Control Plug-in by setting the ***nsslapd-pluginEnabled*** attribute to **on**. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=RootDN Access Control Plug-in,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Set the bind rules for the access control instruction.

   - ***rootdn-open-time*** and ***rootdn-close-time*** for time-based access controls.
   - ***rootdn-days-allowed*** for day-based access controls.
   - ***rootdn-allow-host***, ***rootdn-deny-host***, ***rootdn-allow-ip***, and ***rootdn-deny-ip*** for host-based access controls. These are all multi-valued attributes.

     Deny rules supercede allow rules. For example, if ***rootdn-allow-host*** attribute is set to ***.example.com**, and the ***rootdn-deny-host*** attribute is set to ***.front-office.example.com**, anything in the **front-office.example.com** subdomain is prevented from logging in as Directory Manager, even though the larger **example.com** domain is allowed.

     Wild cards can be used to allow IP ranges or full domains.

   For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=RootDN Access Control Plug-in,cn=plugins,cn=config
changetype: modify
add: rootdn-open-time
rootdn-open-time: 0600
-
add: rootdn-close-time
rootdn-close-time: 2100
-
add: rootdn-allow-host
rootdn-allow-host: *.example.com
-
add: rootdn-deny-host
rootdn-allow-host: *.remote.example.com
```

3. Restart the Directory Server to load the new plug-in configuration.

```
service dirsrv restart instance_name
```

13.12. ACCESS CONTROL AND REPLICATION

ACIs are stored as attributes of entries;therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated like any other attribute.

ACIs are always evaluated on the Directory Server that services the incoming LDAP requests. This means that when a consumer server receives an update request, it returns a referral to the supplier server before evaluating whether the request can be serviced on the supplier.

13.13. COMPATIBILITY WITH EARLIER RELEASES

Some ACI keywords that were used in earlier releases of Directory Server have been deprecated. However, for reasons of backward compatibility, the following keywords are still supported:

- userdnattr
- groupdnattr

Therefore, if you have set up a replication agreement between a legacy supplier server and a version 9.0 consumer, there should not be any problems in the replication of ACIs.

CHAPTER 14. MANAGING USER AUTHENTICATION

When a user connects to the Red Hat Directory Server, first the user is authenticated. Then, the directory grants access rights and resource limits to the user depending upon the identity established during authentication.

This chapter describes tasks for managing users, including configuring the password and account lockout policy for the directory, denying groups of users access to the directory, and limiting system resources available to users depending upon their bind DNs.

14.1. MANAGING THE PASSWORD POLICY

A password policy minimizes the risks of using passwords by enforcing a certain level of security. For example, a password policy can define that:

- Users must change their passwords according to a schedule.
- Users must provide non-trivial passwords.
- The password syntax must meet certain complexity requirements.

For an overview on password policy, see "Designing a Password Policy" in the "Designing a Secure Directory" chapter in the *Deployment Guide*.

WARNING

When using a password administrator account or the **Directory Manager** (root DN) to set a password, password policies are bypassed and not verified. Do not use these accounts for regular user password management. Use them only to perform password administration tasks that require bypassing the password policies.

Directory Server supports fine-grained password policy, so password policies can be applied to the entire directory (*global* password policy), a particular subtree (*subtree-level* or *local* password policy), or a particular user (*user-level* or *local* password policy).

The complete password policy applied to a user account is comprised of the following elements:

- *The type or level of password policy checks.* This information indicates whether the server should check for and enforce a global password policy or local (subtree/user-level) password policies.

  Password policies work in an inverted pyramid, from general to specific. A global password policy is superceded by a subtree-level password policy, which is superceded by a user-level password policy. Only one password policy is enforced for the entry; password policies are not additive. This means that if a particular attribute is configured in the global or subtree-level policy, but not in the user-level password policy, the attribute is not used for the user when a login is attempted because the active, applied policy is the user-level policy.

- *Password add and modify information.* The password information includes password syntax and password history details.

- *Bind information.* The bind information includes the number of grace logins permitted, password aging attributes, and tracking bind failures.

NOTE

After establishing a password policy, user passwords can be protected from potential threats by configuring an account lockout policy. Account lockout protects against hackers who try to break into the directory by repeatedly guessing a user's password.

14.1.1. Configuring the Global Password Policy

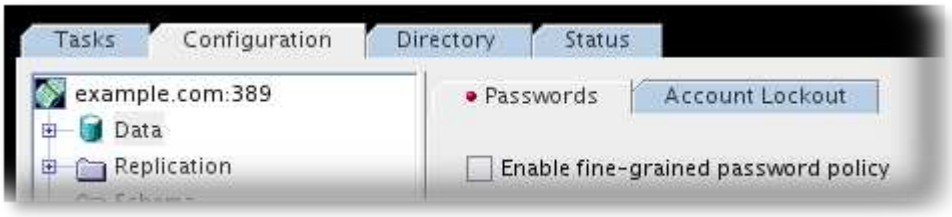
NOTE

After configuring the password policy, configure an account lockout policy. For details, see Section 14.4, “Configuring a Password-Based Account Lockout Policy” .

14.1.1.1. Configuring a Global Password Policy Using the Console

A global password policy applies to every entry in the entire directory.

1. Select the **Configuration** tab and then the **Data** node.
2. In the right pane, select the **Passwords** tab.



   This tab contains the password policy for the entire Directory Server.

3. Set the password policies for how users can change their own passwords.

   - To require users to change their password the first time they log on, select the **User must change password after reset** check box.

     NOTE

     If users are required to reset their password, only the Directory Manager is authorized to reset the user's password. A regular administrative user cannot force the users to update their password.

   - To allow users to change their own passwords, select the **User may change password** check box.
   - To prevent users from changing their password for a specific duration, enter the number of days in the **Allow changes in X day(s)** text box. This keeps users from quickly cycling through passwords to reuse a password in their password history.
   - For the server to maintain a history list of passwords used by each user, select the **Keep password history** check box. Enter the number of passwords for the server to keep for each user in the **Remember X passwords** text box.

4. Set the policies for when passwords expire.

- If user passwords should not expire, select the **Password never expires** radio button.
- To require users to change their passwords periodically, select the **Password expires after X days** radio button, and then enter the number of days that a user password is valid.

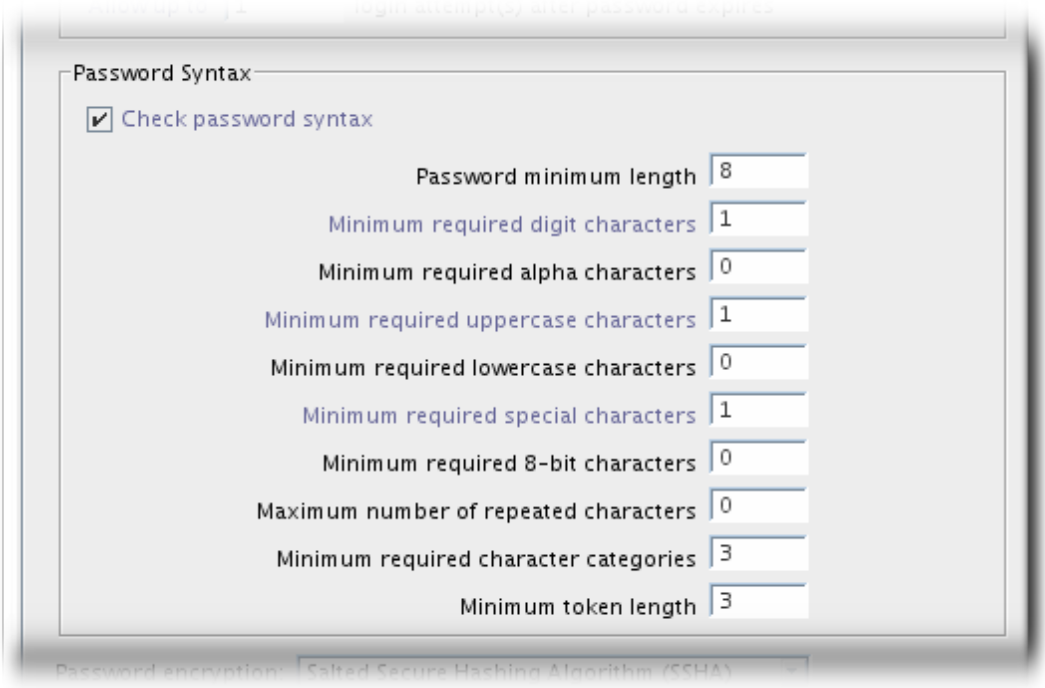
  The maximum value for the password age is derived by subtracting January 18, 2038, from today's date. The entered value must not be set to the maximum value or too close to the maximum value. Setting the value to the maximum value can cause the Directory Server to fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, correct the ***passwordMaxAge*** attribute value in the **dse.ldif** file.

  A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to **8640000** seconds (100 days).
- If the **Password expire after X days** radio button is selected, specify how long before the password expires to send a warning to the user. In the **Send Warning X Days Before Password Expires** text enter the number of days before password expiration to send a warning.

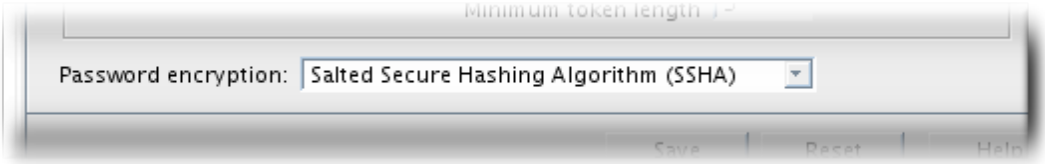
NOTE

It is not necessary to configure the Directory Server to send a warning to users. The Directory Server automatically issues a warning the next time the user attempts to log into the Directory Server Console that the password will soon expire or has expired. This is analogous to an operating system warning that reads **"Warning: password will expire in 7 days"** when a user logs in.

5. For the server to check the syntax of a user password to make sure it meets the minimum requirements set by the password policy, select the **Check Password Syntax** check box. Then, specify required password complexity, such as the minimum length and required number of numeric and special characters. The password syntax requirements are described more in Table 14.1, “Password Policy Attributes”.



6. From the **Password Encryption** pull-down menu, select the encryption method for the server to use when storing passwords.



   For detailed information about the encryption methods, see the ***passwordStorageScheme*** attribute in Table 14.1, "Password Policy Attributes".

   The **Password Encryption** menu might contain other encryption methods, as the directory dynamically creates the menu depending upon the existing encryption methods it finds in the directory.

7. Click **Save**.

14.1.1.2. Configuring a Global Password Policy Using the Command Line

To set up the password policy for a subtree or user, add the required entries and attributes at the subtree- or user-level, set the appropriate values to the password policy attributes, and enable fine-grained password policy checking.

No password policy attributes are set by default. Each password policy attribute must be added manually to the **cn=config** entry to create a global policy. These can be passed all together by passing an LDIF file with **ldapmodify**.

1. Create the LDIF file. Each statement is the same as inputting the changes through stdin, with separate update statements separated by a dash (**-**).

```
dn: cn=config
changetype: modify
add: passwordChange
passwordChange: on
-
add: passwordExp
passwordExp: on
-
add: passwordMaxAge
passwordMaxAge: 8640000
-
add: passwordCheckSyntax
passwordCheckSyntax: on
-
```

```
add: passwordMinCategories
passwordMinCategories: 3
-
add: passwordStorageScheme
passwordStorageScheme: SSHA512
^D
```

2. Pass the LDIF file to the server using the **-f** option with the **ldapmodify** command.

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -f user-pwdpolicy.ldif
```

Table 14.1, "Password Policy Attributes" describes the attributes available to configure the password policy.

Table 14.1. Password Policy Attributes

Attribute Name	Definition
passwordTrackUpdateTime	This attribute sets whether to record a separate timestamp specifically for the most recent update to the password attribute. This can be useful to help coordinate sycnhronization updates or changes between other LDAP servers, like Active Directory. By default, this is turned off.
passwordGraceLimit	This attribute indicates the number of grace logins permitted when a user's password is expired. When set to a positive number, the user will be allowed to bind with the expired password for that many times. For the global password policy, the attribute is defined under **cn=config**. By default, this attribute is set to **0**, which means grace logins are not permitted.
passwordMustChange	When **on**, this attribute requires users to change their passwords when they first login to the directory or after the password is reset by the Directory Manager. The user is required to change their password even if user-defined passwords are disabled. If this attribute is set to **off**, passwords assigned by the Directory Manager should not follow any obvious convention and should be difficult to discover. This attribute is **off** by default.
passwordChange	When **on**, this attribute indicates that users may change their own password. Allowing users to set their own passwords runs the risk of users choosing passwords that are easy to remember. However, setting good passwords for the user requires a significant administrative effort. In addition, providing passwords to users that are not meaningful to them runs the risk that users will write the password down somewhere that can be discovered. This attribute is **on** by default.
passwordExp	When **on**, this attribute indicates that the user's password will expire after an interval given by the ***passwordMaxAge*** attribute. Making passwords expire helps protect the directory data because the longer a password is in use, the more likely it is to be discovered. This attribute is **off** by default.

Attribute Name	Definition
passwordMaxAge	This attribute indicates the number of seconds after which user passwords expire. To use this attribute, enable password expiration using the ***passwordExp*** attribute. This attribute is a dynamic parameter in that its maximum value is derived by subtracting January 18, 2038, from today's date. The attribute value must not be set to the maximum value or too close to the maximum value. If the value is set to the maximum value, Directory Server may fail to start because the number of seconds will go past the epoch date. In such an event, the error log will indicate that the password maximum age is invalid. To resolve this problem, correct the ***passwordMaxAge*** attribute value in the **dse.ldif** file. A common policy is to have passwords expire every 30 to 90 days. By default, the password maximum age is set to **8640000** seconds (100 days).
passwordWarning	This attribute indicates the number of seconds before a warning message is sent to users whose password is about to expire. Depending on the LDAP client application, users may be prompted to change their password when the warning is sent. By default, the directory sends the warning **86400** seconds (1 day) before the password is about to expire. However, a password never expires until the warning message has been sent. Therefore, if users do not bind to the Directory Server for longer than the ***passwordMaxAge***, they will still get the warning message in time to change their password.
passwordMinAge	This attribute indicates the number of seconds that must pass before a user can change their password. Use this attribute in conjunction with the ***passwordInHistory*** attribute to discourage users from reusing old passwords. For example, setting the minimum password age to 2 days prevents users from repeatedly changing their passwords during a single session to cycle through the password history and reuse an old password once it has been removed from the history list. The minimum age can be from **0** to **2147472000** seconds (24,855 days). A value of zero indicates that the user can change the password immediately. The default value of this attribute is **0**.
passwordHistory	This attribute indicates whether the directory stores a password history. When set to **on**, the directory stores the number of passwords specified in the ***passwordInHistory*** attribute in a history. If a user attempts to reuse one of the passwords, the password will be rejected. When this attribute is set to **off**, any passwords stored in the history remain there. When this attribute is set back to **on**, users will not be able to reuse the passwords recorded in the history before the attribute was disabled. This attribute is **off** by default, meaning users can reuse old passwords.
passwordInHistory	This attribute indicates the number of passwords the directory stores in the history. There can be **2** to **24** passwords stored in the history. This feature is not enabled unless the ***passwordHistory*** attribute is set to **on**. This attribute is set to **6** by default.

Attribute Name	Definition
passwordCheckSyntax	When **on**, this attribute indicates that the password syntax is checked by the server before the password is saved. Password syntax checking ensures that the password string meets or exceeds the length and complexity requirements and that the string does not contain any *trivial* words. A trivial word is any value stored in the ***uid***, ***cn***, ***sn***, ***givenname***, ***ou***, or ***mail*** attributes of the user's entry. This attribute is **off** by default.
passwordMinLength	This attribute specifies the minimum number of characters that must be used in passwords. Shorter passwords are easier to crack. Passwords can be two (2) to 512 characters long. Generally, a length of eight characters is long enough to be difficult to crack but short enough for users to remember without writing it down. This attribute is set to **8** by default.
passwordMaxRepeats	This attribute set the maximum number of times that the same character can be used in row, such as **aaaaa**. Setting the attribute to **0** means that there is no limit on how many time a character can be repeated. This attribute is set to **0** by default.
passwordMinAlphas	This attribute sets the minimum number of alphabetic characters that must be used in the password. This setting does not set any requirements for the letter case; requirements for the minimum number of lowercase and uppercase letters are set in the ***passwordMinLower*** and ***passwordMinUpper*** attributes, respectively. By default, this attribute is set to **0**, meaning there is no required minimum.
passwordMinDigits	This attribute sets the minimum number of numeric characters (0 through 9) which must be used in the password. By default, this attribute is set to **0**, meaning there is no required minimum.
passwordMinSpecials	This attribute sets the minimum number of special ASCII characters, such as **!@#$.**, which must be used in the password. By default, this attribute is set to **0**, meaning there is no required minimum.
passwordMinLowers	This attribute sets the minimum number of lower case alphabetic characters, a to z, which must be used in the password. By default, this attribute is set to **0**, meaning there is no required minimum.
passwordMinCategories	This attribute sets the minimum number of categories which must be used in the password. There are several categories available: Uppercase letters (A to Z) Lowercase letters (a to z) Numbers (0 through 9) Special ASCII characters, such as **$** 8-bit characters This attribute is set to **3** by default.

Attribute Name	Definition
passwordMinUppers	This attribute sets the minimum number of upper case alphabetic characters, A to Z, which must be used in the password. By default, this attribute is set to **0**, meaning there is no required minimum.
passwordMinTokenLength	A trivial word check identifies dictionary or common words used in a password string. This attribute sets the minimum length for tokens or strings which are checked by the ***passwordSyntaxCheck*** attributes. This can be from**1** to **64** characters. This attribute is set to**3** by default, meaning that the password cannot contain a three-character (or longer) sequential string taken from their user name, common name, first or last name, mail, or other relevant attributes. For example, for a token length of three, if the user name is **jsmith**, then the password**123smi!** would be rejected because it has three sequential characters which match the user name.
passwordMin8bit	This attribute sets the minimum number of 8-bit characters used in the password. The default number is **0**, meaning none are required.
passwordStorageScheme	This attribute specifies the type of encryption used to store Directory Server passwords. The following encryption types are supported by Directory Server: *SSHA (Salted Secure Hash Algorithm).* This method is recommended as it is the most secure. The Directory Server supports **SSHA**, **SSHA256**, **SSHA384**, and **SSHA512**. SSHA is the default method. *SHA (Secure Hash Algorithm).* A one-way hash algorithm; it is supported only for backwards compatibility with Directory Server 4.x and should not be used otherwise. This includes support for **SHA**, **SHA256**, **SHA384**, and **SHA512** algorithms, which protects against some insecurities in the SHA1 algorithm. *MD5.* MD5 is not as secure as SSHA but is available for legacy applications require it. *Salted MD5.* This storage scheme is more secure than plain MD5 hash, but still less secure than SSHA. This storage scheme is not included for use with new passwords but to help with migrating user accounts from directories which support salted MD5. *crypt.* The UNIX crypt algorithm, provided for compatibility with UNIX passwords. *clear.* This encryption type indicates that the password will appear in plain text. The only password storage scheme that can be used with SASL DIGEST-MD5 is **CLEAR**. Passwords stored using **crypt**, **SHA**, or **SSHA** formats cannot be used for secure login through SASL DIGEST-MD5. To provide a customized storage scheme, consult Red Hat professional services.

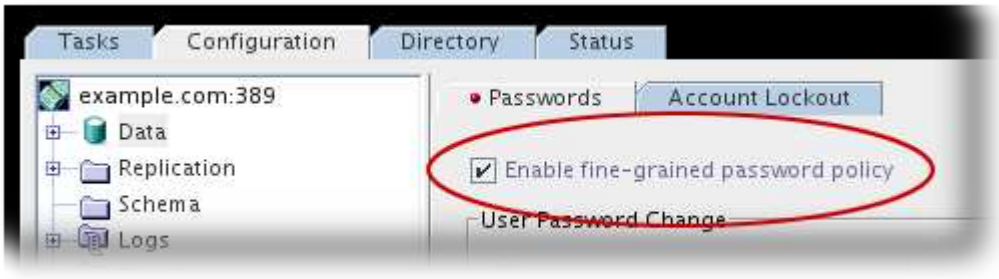
14.1.2. Configuring a Local Password Policy

NOTE

After configuring the password policy, configure an account lockout policy. For details, see Section 14.4, "Configuring a Password-Based Account Lockout Policy" .

14.1.2.1. Configuring a Subtree/User Password Policy Using the Console

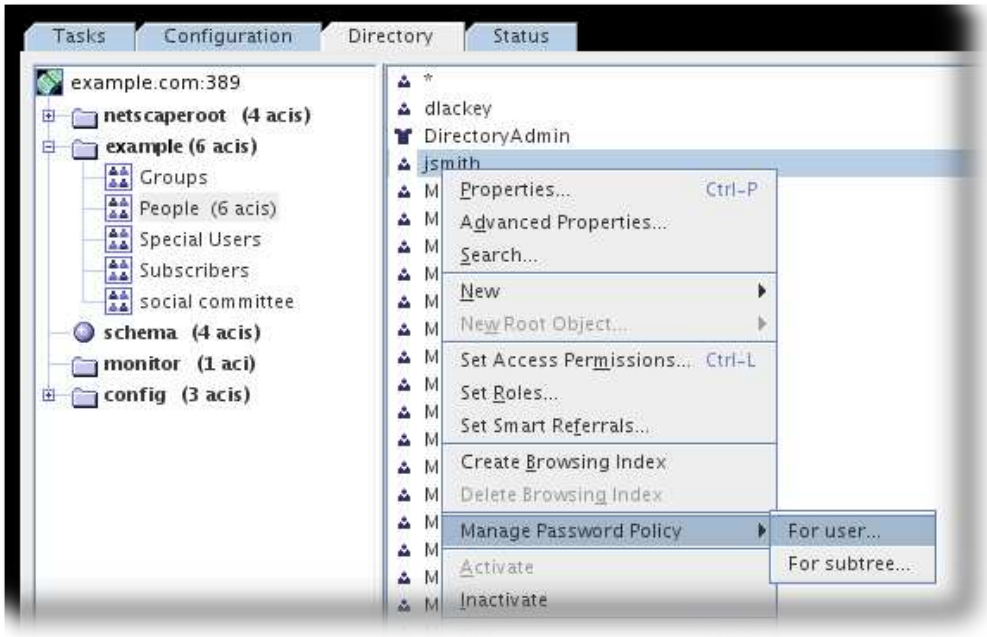
1. Enable a fine-grained password policy globally, as described in Section 14.1.1.1, "Configuring a Global Password Policy Using the Console". Be sure to check the **Enable fine-grained password policy** check box to allow user-level password policies.



NOTE

The password policy *must* be enabled globally before it will be applied locally. No other global password policy features must be set, and the global password policy will not override the local policy if they differ.

2. Create the local password policy for the subtree or user.
    1. Select the **Directory** tab.
    2. In the navigation pane, select the subtree or user entry for which to set up the password policy.
    3. From the **Object** menu, select the **Manage Password Policy** option, and then select the **For user** or **For subtree**.



    4. In the **Passwords** tab, select the **Create subtree/user level password policy** check box to add the required attributes. The password policy settings — resetting, expiration, syntax, and encryption — are the same as for the global policy in Section 14.1.1.1, "Configuring a Global Password Policy Using the Console" .

5. In the **Account Lockout** tab, specify the appropriate information, and click **Save**.

14.1.2.2. Configuring Subtree/User Password Policy Using the Command Line

1. Add the required attributes to the subtree or user entries by running the **ns-newpwpolicy.pl** script.

   The command syntax for the script is as follows:

   ```
   ns-newpwpolicy.pl [-D rootDN]  -w password | -w - | -j filename [-p port] [-h host] -U userDN -S suffixDN
   ```

   For updating a subtree entry, use the **-S** option. For updating a user entry, use the **-U** option. The **ns-newpwpolicy.pl** script accepts only one user or subtree entry at a time. It can, however, accept both user and suffix entries at the same time. For details about the script, see the *Directory Server Configuration and Command-Line Tool Reference*.

2. The script adds the required attributes depending on whether the target entry is a subtree or user entry.

   For a subtree (for example, **ou=people,dc=example,dc=com**), the following entries are added:

   - A container entry (***nsPwPolicyContainer***) at the subtree level for holding various password policy-related entries for the subtree and all its children. For example:

     ```
     dn: cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
     objectClass: top
     objectClass: nsContainer
     cn: nsPwPolicyContainer
     ```

   - The actual password policy specification entry (***nsPwPolicyEntry***) for holding all the password policy attributes that are specific to the subtree. For example:

     ```
     dn: cn=cn=nsPwPolicyEntry\,ou=people\,dc=example\,dc=com,
         cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
     objectclass: top
     ```

```
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

- The CoS template entry (***nsPwTemplateEntry***) that has the ***pwdpolicysubentry*** value pointing to the above (***nsPwPolicyEntry***) entry. For example:

```
dn: cn=cn=nsPwTemplateEntry\,ou=people\,dc=example\,dc=com,
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: costemplate
objectclass: ldapsubentry
cosPriority: 1
pwdpolicysubentry: cn=cn=nsPwPolicyEntry\,ou=people\,dc=example\,dc=com,
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
```

- The CoS specification entry at the subtree level. For example:

```
dn: cn=newpwdpolicy_cos,ou=people,dc=example,dc=com
objectclass: top
objectclass: LDAPsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=cn=nsPwTemplateEntry\,ou=people\,dc=example,dc=com,
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
cosAttribute: pwdpolicysubentry default operational
```

For a user (for example, **uid=jdoe,ou=people,dc=example,dc=com**), the following entries are added:

- A container entry (***nsPwPolicyContainer***) at the parent level for holding various password policy related entries for the user and all its children. For example:

```
dn: cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectClass: top
objectClass: nsContainer
cn: nsPwPolicyContainer
```

- The actual password policy specification entry (***nsPwPolicyEntry***) for holding the password policy attributes that are specific to the user. For example:

```
dn: cn=cn=nsPwPolicyEntry\,uid=jdoe\,ou=people\,dc=example\,dc=com,
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
objectclass: top
objectclass: extensibleObject
objectclass: ldapsubentry
objectclass: passwordpolicy
```

3. Assign the value of the above entry DN to the ***pwdpolicysubentry*** attribute of the target entry. For example, this assigns the password policy to the user entry:

```
dn: uid=jdoe,ou=people,dc=example,dc=com
changetype: modify
replace: pwdpolicysubentry
pwdpolicysubentry: cn=cn=nsPwPolicyEntry\,uid=jdoe\,ou=people\,dc=example\,dc=com,
    cn=nsPwPolicyContainer,ou=people,dc=example,dc=com
```

4. Set the password policy attributes for the subtree or user entry with the appropriate values.

   Table 14.1, “Password Policy Attributes” describes the attributes available to configure the password policy. The **ldapmodify** utility can be used to change these attributes in the subtree or user entry which contains the **nsPwPolicyEntry** object class.

NOTE

The ***nsslapd-pwpolicy-local*** attribute of the **cn=config** entry controls the type of password policy the server enforces. By default, this attribute is disabled (**off**). When the attribute is disabled, the server only checks for and enforces the global password policy; the subtree and user-level password policies are ignored. When the **ns-newpwpolicy.pl** script runs, it first checks for the specified subtree and user entries and, if they exist, modifies them. After updating the entries successfully, the script sets the ***nsslapd-pwpolicy-local*** configuration parameter to on. If the subtree and user-level password policy should not be enabled, be sure to set ***nsslapd-pwpolicy-local*** to **off** after running the script.

To turn off user- and subtree-level password policy checks, set the ***nsslapd-pwpolicy-local*** attribute to **off** by modifying the **cn=config** entry. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-pwpolicy-local
nsslapd-pwpolicy-local: off
```

This attribute can also be disabled by modifying it directly in the configuration file (**dse.ldif**).

1. Stop the server.

   ```
   service dirsrv stop instance
   ```

2. Open the **dse.ldif** file in a text editor.

3. Set the value of ***nsslapd-pwpolicy-local*** to **off**, and save.

   ```
   nsslapd-pwpolicy-local: off
   ```

4. Start the server.

   ```
   service dirsrv start instance
   ```

14.1.2.3. Manually Setting Default Password Syntax Checking for Local Password Policies

The settings for the global password policy should be in effect for any local password policies, if those attributes are not explicitly set. If the password policy attribute is not set in either the global or the local policy, then the default values should be assumed.

However, there is a bug in Directory Server, so that if a password policy attribute is set in the global password policy but not in the local password policy, then neither the global setting nor the default settings is enforced by the local password policy. To work around this, set the password attributes explicitly in the local password policy.

1. Enable global syntax checking, as in Section 14.1.1.1, “Configuring a Global Password Policy Using the Console”, and save the policy.

2. Edit the local password policy to contain all password syntax

3. Enable fine-grained password checking, as in Section 14.1.2.1, “Configuring a Subtree/User Password Policy Using the Console”, and save the policy.

4. Edit the local password policy to contain all password syntax attributes. Set the values to something other than the default settings, as listed in the *Configuration and Command-Line Tool Reference* .

5. Re-edit the local password policy with the desired values, even if they are the defaults.

14.1.3. Setting User Passwords

An entry can be used to bind to the directory only if it has a ***userPassword*** attribute and if it has not been inactivated. Because user passwords are stored in the directory, the user passwords can be set or reset with any LDAP operation, like **ldapmodify**.

For information on creating and modifying directory entries, see Chapter 3, *Creating Directory Entries*. For information on inactivating user accounts, see Section 14.11, “Manually Inactivating Users and Roles” .

Passwords can also be set and reset in the **Users and Groups** area of the Red Hat Admin Server or Directory Server Console. For information on how to use the **Users and Groups** area in the Admin Server Console, see the online help that is available in the Red Hat Admin Server.

Only password administrators, described in Section 14.1.4, "Setting Password Administrators", and the root DN can add pre-hashed passwords. These users can also violate password policy.

WARNING

When using a password administrator account or the **Directory Manager** (root DN) to set a password, password policies are bypassed and not verified. Do not use these accounts for regular user password management. Use them only to perform password administration tasks that require bypassing the password policies.

14.1.4. Setting Password Administrators

The Directory Manager can add the *password administrator* role to a user or a group of users. Since access control instructions (ACI) need to be set, it is recommended that a group is used to allow just a single ACI set to manage all password administrators. A password administrator can perform any user password operations, including the following:

- forcing the user to change their password,
- changing a user's password to a different storage scheme defined in the password policy,
- bypassing the password syntax checks,
- and adding already hashed passwords.

As explained in Section 14.1.3, "Setting User Passwords", it is recommended that ordinary password updates are done by an existing role in the database with permissions to update only the **userPassword** attribute. We recommend not to use the password administrator account for these ordinary tasks.

To specify a user or a group of users as password administrator in a local policy, use **ldapmodify** to set the **passwordAdminDN** attribute in the main configuration entry.

```
[root@server ~]# ldapmodify -h localhost -p 389 -D "cn=directory manager" -W
dn:
cn=cn\3DnsPwPolicyEntry\2Cou\3DPeople\2Cdc\3Dexample\2Cdc\3Dcom,cn=nsPwPolicyContainer,ou=People,dc=e
xample,dc=com
changetype: modify
replace: passwordAdminDN
passwordAdminDN: cn=Passwd Admins,ou=groups,dc=example,dc=com
```

For setting in the global policy:

```
[root@server ~]# ldapmodify -h localhost -p 389 -D "cn=directory manager" -W
dn: cn=config
changetype: modify
replace: passwordAdminDN
passwordAdminDN: cn=Passwd Admins,ou=groups,dc=example,dc=com
```

14.1.5. Changing Passwords Stored Externally

While most passwords can be changed through the Console and other Directory Server features or through the **ldapmodify** operation, there are some passwords that cannot be changed through regular LDAP operations. These passwords may be stored outside the Directory Server, such as passwords stored in a SASL application. These passwords can be modified through the *password change extended operation*.

Directory Server supports the password change extended operation as defined in RFC 3062, so users can change their passwords, using a suitable client, in a standards-compliant way. The **ldappasswd** utility passes the changes for the password for the specified user:

```
ldappasswd -x -D bind_dn -w password -p server_port -h server_hostname [-a oldPassword] [-s newPassword]
[user]
```

IMPORTANT

Password operations must be performed over a secure connection, meaning SASL, TLS/SSL, or Start TLS. For information on using secure connections with LDAP client tools, see Section A.2, "Using SSL/TLS and Start TLS with LDAP Client Tools".

Table 14.2. ldappasswd Options

Parameter	Description
-h	Gives the host name of the Directory Server.
-p	Gives the port number of the Directory Server. Since SSL is required for password change operations, this is usually give the TLS/SSL port of the Directory Server. With the **-ZZ** or **-ZZZ** for Start TLS, this can be the standard port.
-D	Gives the bind DN.
-w	Gives the password for the bind DN.
-x	Disables SASL to allow a simple bind over an SSL/TLS connection.
-a	*Optional.* Gives the old password, which is being changed.
-s	*Optional.* Sets the new password.
user	*Optional.* Gives the DN of the user entry for which to change the password.

To use Start TLS, which runs the command on a non-secure port, run **ldappasswd** with the **-ZZ** option and the standard LDAP port number. The password extended change operation has the following format:

```
ldappasswd -x -D bind_dn -w password -p server_port -h server_hostname -Z  [-a oldPassword] [-s newPassword] [user]
```

NOTE

For Start TLS connections to work, the SSL/TLS environment variables must be configured as described in Section A.2, "Using SSL/TLS and Start TLS with LDAP Client Tools" .

Use the **-ZZ** to force the connection to be successful.

To modify an entry's password, run **ldappasswd** like any other LDAP operation. It is not necessary to specify a *user* if the account is the same as that given in the bind DN. For example:

```
ldappasswd -x -h ldap.example.com -p 389 -ZZ -D "uid=jsmith,ou=People,dc=example,dc=com" -W -s newpassword
```

To change the password on an entry other than the one specified in the bind credentials, run **ldappasswd** as shown below, adding the *user* DN to the operation and providing separate credentials, as follows:

```
ldappasswd -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -ZZ -W -s newpassword "uid=jsmith,ou=People,dc=example,dc=com"
```

Access control is enforced for the password change operation. If the bind DN does not have rights to change the specified password, the operation will fail with an **Insufficient rights** error.

14.2. MANAGING THE DIRECTORY MANAGER PASSWORD

The Directory Manager is the privileged database administrator, comparable to the **root** user in UNIX. The Directory Manager entry and its associated password are created during installation. The default DN is **cn=Directory Manager**.

14.2.1. Changing the Directory Manager Password

The password for the Directory Manager superuser is defined in the ***nsslapd-rootpw*** attribute.

NOTE

The password can be changed using **ldapmodify** by sending the password in plaintext. Changing the password through the Directory Server Console ensures that the password is immediately hashed when it is saved in the **dse.ldif** file, so it is never saved in clear text.

IMPORTANT

When resetting the Directory Manager's password from the command line, *do not* use curly braces (**{}**) in the password. The root password is stored in the format *{password-storage-scheme}hashed_password*. Any characters in curly braces are interpreted by the server as the password storage scheme for the root password. If that text is not a valid storage scheme or if the password that follows is not properly hashed, then the Directory Manager cannot bind to the server.

NOTE

Both the Directory Manager password and its password storage scheme can be changed in the Directory Server Console. The password by itself can be changed. However, if the password storage scheme is changed than the password must also be changed, so that it can be rehashed and stored with the new scheme.

1. Log into the Directory Server Console as Directory Manager.
2. Select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
3. Select the **Manager** tab in the right pane.

4. Enter a new password, and confirm it.

14.2.2. Changing the Directory Manager Password Storage Scheme

The ***nsslapd-rootpw*** attribute contains a hash of the password and an indication of the password storage scheme in braces, such as **{SSHA}**. If the password is in clear text, then the password storage scheme is CLEAR. The default password storage scheme is SSHA.

```
nsslapd-rootpw: {SSHA}od1V7JmQlMdldxrOlp3XSnMuXZVsXi8/YUVM7Q==
nsslapd-rootpwstoragescheme: SSHA
```

To change the password storage scheme:

1. Log into the Directory Server Console as Directory Manager. For information on changing the bind DN, see Section 1.4.3, “Changing the Login Identity”.
2. Select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.
3. Select the **Manager** tab in the right pane.

4. Set the storage scheme for the server to use to store the password for Directory Manager in the **Manager Password Encryption** pull-down menu.

5. Enter a new password, and confirm it.

NOTE

If the password storage scheme is changed, then the password must also be changed so that it can be rehashed and stored with the new scheme.

As always, *do not* use curly braces (**{}**) in the password. The root password is stored in the format *{password-storage-scheme}hashed_password*. Any characters in curly braces are interpreted by the server as the password storage scheme for the root password. If that text is not a valid storage scheme or if the password that follows is not properly hashed, then the Directory Manager cannot bind to the server.

14.2.3. Changing the Directory Manager DN

The default DN for the Directory Manager is **cn=Directory Manager**, which is created when the Directory Server is installed. This DN can be changed to a unique DN.

1. Log into the Directory Server Console as Directory Manager.

2. Select the **Configuration** tab, and then select the top entry in the navigation tree in the left pane.

3. Select the **Manager** tab in the right pane.

4. Change the distinguished name for the Directory Manager in the **Root DN** field. The default value is **cn=Directory Manager**.

14.3. CHECKING ACCOUNT AVAILABILITY FOR PASSWORDLESS ACCESS

Most of the time, for the Directory Server to return authentication information about a user account, a client actually binds (or attempts to bind) as that user. And a bind attempt requires some sort of user credentials, usually a password or a certificate. While the Directory Server allows unauthenticated binds and anonymous binds, neither of those binds returns any user account information.

There are some situations where a client requires information about a user account – specifically whether an account should be allowed to authenticate – in order to perform some other operation, but the client either does not have or does use any credentials for the user account in Directory Server. Essentially, the client needs to perform a credential-less yet authenticated bind operation to retrieve the user account information (including password expiration information, if the account has a password).

This can be done through an **ldapsearch** by passing the *Account Usability Extension Control*. This control acts as if it performs an authenticated bind operation for a given user and returns the account status for that user — but without actually binding to the server. This allows a client to determine whether that account can be used to log in and then to pass that account information to another application, like PAM.

For example, using the Account Usability Extension Control can allow a system to use the Directory Server as its identity back end to store user data but to employ password-less authentication methods, like smart cards or SSH keys, where the authentication operation is performed outside Directory Server.

14.3.1. Searching for Entries Using the Account Usability Extension Control

The Account Usability Extension Control is an extension for an **ldapsearch**. It returns an extra line for each returned entry that gives the account status and some information about the password policy for that account. A client or application can then use that status to evaluate authentication attempts made outside Directory Server for that user account. Basically, this control signals whether a user should be allowed to authenticate without having to perform an authentication operation.

NOTE

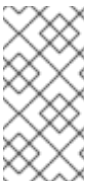
The OpenLDAP tools used by Directory Server do not support the Account Usability Extension Control. Other LDAP utilities, like OpenDS, can be used or other clients which do support the control.

For example, using the OpenDS tools, the control can be specified using the **-J** with the control OID (1.3.6.1.4.1.42.2.27.9.5.8) or with the **accountusability:true** flag:

```
[jsmith@server ~]$ pathToOpenDsLdapTools/bin/ldapsearch -D "cn=directory manager" -W -p 389 -h
server.example.com -b "dc=example,dc=com" -s sub -J "accountusability:true" "(objectclass=*)"
# Account Usability Response Control
# The account is usable
dn: dc=example,dc=com
objectClass: domain
objectClass: top
dc: example
...
```

This can also be run for a specific entry:

```
[jsmith@server ~]$ pathToOpenDsLdapTools/bin/ldapsearch -D "cn=directory manager" -W -p 389 -h
server.example.com -b "uid=bjensen,ou=people,dc=example,dc=com" -s base -J "accountusability:true" "
(objectclass=*)"
# Account Usability Response Control
# The account is usable
dn: uid=bjensen,ou=people,dc=example,dc=com
...
```



NOTE

By default, only the Directory Manager can use the Account Usability Extension Control. To allow other users to use the Account Usability Extension Control, set on ACI on the supported control entry under **cn=features**. See Section 14.3.2, “Changing What Users Can Perform an Account Usability Search”.

The control returns different messages, depending on the actual status of the account and (if the user has a password) the password policy settings for the user account.

Table 14.3. Possible Account Usability Control Result Messages

Account Status	Control Result Message
Active account with a valid password	The account is usable
Active account with no password set	The account is usable
Expired password	Password expired
The password policy for the account is modified	Password expired

Account Status	Control Result Message
The account is locked and there is no lockout duration	Password reset
The account is locked and there is a lockout duration	*Time* (in seconds) for automatic unlock of the account
The password for the account should be reset at the first login	Password reset
The password has expired and grace logins are allowed	Password expired and *X* grace login is allowed
The password has expired and the number of grace logins is exhausted	Password expired
The password will expire (expiration warning)	Password will expire in *X* number of seconds

14.3.2. Changing What Users Can Perform an Account Usability Search

By default, only the Directory Manager can use the Account Usability Extension Control. Other users can use the Account Usability Extension Control by setting the appropriate ACI on the supported control entry. The control entry is named for the Account Usability Extension Control OID, 1.3.6.1.4.1.42.2.27.9.5.8.

For example:

```
[jsmith@server ~]$ ldapmodify -D "cn=directory manager" -W -x

dn: oid=1.3.6.1.4.1.42.2.27.9.5.8,cn=features,cn=config
changetype: modify
add: aci
aci: (targetattr != "aci")(version 3.0; acl "Account Usable"; allow (read, search, compare, proxy)(groupdn =
"ldap:///cn=Administrators,ou=groups,dc=example,dc=com");)
```

14.3.3. Using pam_ldap Account Management with the Account Usability Control

The **pam_ldap** module allows a local system to use passwordless authentication mechanisms (like SSH keys, rlogin, and rsh) while still using an LDAP directory to store user information. The **pam_ldap** module connects to the Directory Server to verify that an account is active, is not locked, and has a valid password (if applicable). When a user uses a passwordless authentication method, then the **pam_ldap** module must retrieve the account status without authenticating to the LDAP server as the actual user.

PAM can be configured first for system account management, and then to use the **pam_ldap** module as one of its authentication sources. On Red Hat Enterprise Linux, the PAM account management is configured in the **/etc/pam.d/other** file.

The default account configuration does not use PAM for account management:

```
account  required       pam_deny.so
```

This can be changed to use first system accounts and then LDAP-stored accounts for system authentication. For example:

```
account  requisite      pam_roles.so.1
account  binding        pam_unix_account.so.1 server_policy
account  required       pam_ldap.so.1
```

If a user is found in a system account, that account is used first (**pam_unix_account**). If the user is not found there, then PAM checks the LDAP server. The LDAP configuration is set in the **/etc/pam_ldap.conf** file. That configuration file needs to set only the connection information for the Directory Server instance, such as its base DN, host, and port. Because the Account Usability Plug-in is enabled by default, the Directory Server is already configured to allow Account Usability Control searches for clients.

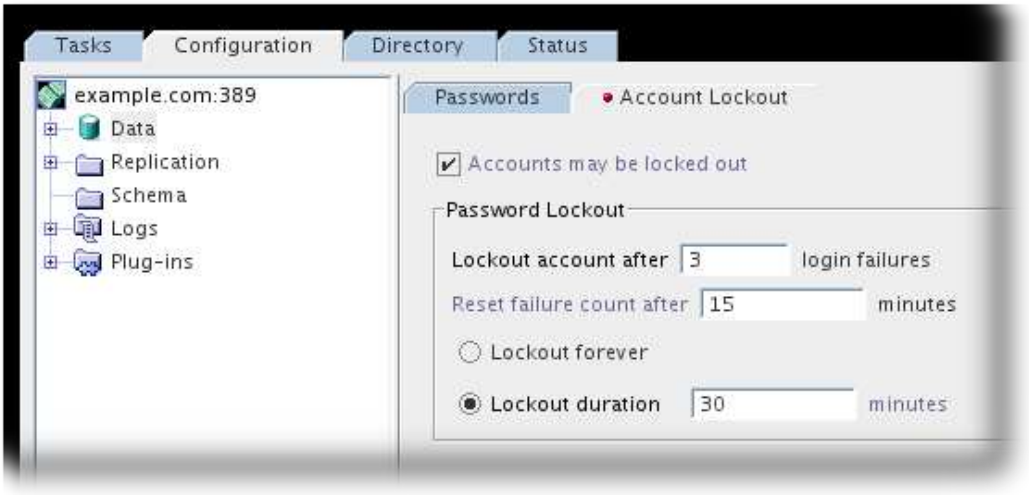
14.4. CONFIGURING A PASSWORD-BASED ACCOUNT LOCKOUT POLICY

A password-based account lockout policy protects against hackers who try to break into the directory by repeatedly trying to guess a user's password. The password policy can be set so that a specific user is locked out of the directory after a given number of failed attempts to bind.

14.4.1. Configuring the Account Lockout Policy Using the Console

To set up or modify the account lockout policy for the Directory Server:

1. Select the **Configuration** tab and then the **Data** node.
2. In the right pane, select the **Account Lockout** tab.



3. To enable account lockout, select the **Accounts may be locked out** check box.
4. Enter the maximum number of allowed bind failures in the **Lockout account after X login failures** text box. The server locks out users who exceed the limit specified here.
5. In the **Reset failure counter after X minutes** text box, enter the number of minutes for the server to wait before resetting the bind failure counter to zero.
6. Set the interval for users to be locked out of the directory.
   - Select the **Lockout Forever** radio button to lock users out until their passwords have been reset by the administrator.
   - Set a specific lockout period by selecting the **Lockout Duration** radio button and entering the time (in minutes) in the text box.
7. Click **Save**.

14.4.2. Configuring the Account Lockout Policy Using the Command Line

Table 14.4, "Account Lockout Policy Attributes" describes the attributes available to configure the account lockout policy. Use **ldapmodify** to change these attributes in the **cn=config** entry. For example:

```
[jsmith@server ~]$ ldapmodify -D "cn=directory manager" -W -x -p 389 -h server.example.com -x
dn: cn=config
changetype: modify
replace: passwordLockout
passwordLockout: on
-
add: passwordMaxFailure
passwordMaxFailure: 4
-
add: passwordLockoutDuration
passwordLockoutDuration: 600
-
```

Table 14.4. Account Lockout Policy Attributes

Attribute Name	Definition
passwordLockout	This attribute indicates whether users are locked out of the directory after a given number of failed bind attempts. Set the number of failed bind attempts after which the user will be locked out using the ***passwordMaxFailure*** attribute. Users can be locked out for a specific time or until an administrator resets the password. This attribute is set to **off** by default, meaning that users will not be locked out of the directory.
passwordMaxFailure	This attribute indicates the number of failed bind attempts after which a user will be locked out of the directory. This attribute takes affect only if the ***passwordLockout*** attribute is set to **on**. This attribute is set to**3** bind failures by default.
passwordUnlock	This attribute sets whether a user can log back into the server without administrator intervention. The default is for this attribute to be on, meaning that the user *can* log back into the server after a certain lockout period has passed. If
passwordLockoutDuration	This attribute indicates the time, in seconds, that users will be locked out of the directory. The ***passwordUnlock*** attribute specifies if a user is locked out until the password is reset by an administrator (which means that the user is locked out indefinitely). If the ***passwordUnlock*** attribute is set to **on**, then the use can log in again as soon as the lockout duration time is reached. By default, the user is locked out for 3600 seconds.
passwordResetFailureCount	This attribute specifies the time, in seconds, after which the password failure counter will be reset. Each time an invalid password is sent from the user's account, the password failure counter is incremented. If the ***passwordLockout*** attribute is set to **on**, users will be locked out of the directory when the counter reaches the number of failures specified by the ***passwordMaxFailure*** attribute. The account is locked out for the interval specified in the ***passwordLockoutDuration*** attribute, after which time the failure counter is reset to zero (**0**). Because the counter's purpose is to gage when a hacker is trying to gain access to the system, the counter must continue for a period long enough to detect a hacker. However, if the counter were to increment indefinitely over days and weeks, valid users might be locked out inadvertently. The reset password failure count attribute is set **600** seconds by default.

14.4.3. Disabling Legacy Password Lockout Behavior

There are different ways of interpreting when the maximum password failure (***passwordMaxFailure***) has been reached. It depends on how the server counts the last failed attempt in the overall failure count.

The traditional behavior for LDAP clients is to assume that the failure occurs *after* the limit has been reached. So, if the failure limit is set to three, then the lockout happens at the fourth failed attempt. This also means that if the fourth attempt is successful, then the user can authenticate successfully, even though the user technically hit the failure limit. This is *n+1* on the count.

LDAP clients increasingly expect the maximum failure limit to look at the last failed attempt in the count as the final attempt. So, if the failure limit is set to three, then at the third failure, the account is locked. A fourth attempt, even with the correct credentials, fails. This is *n* on the count.

The first scenario – where an account is locked only if the attempt count is exceeded – is the historical behavior, so this is considered a legacy password policy behavior. In Directory Server, this policy is enabled by default, so an

account is only locked when the failure count is *n+1*. This legacy behavior can be disabled so that newer LDAP clients receive the error (LDAP_CONSTRAINT_VIOLATION) when they expect it. This is set in the ***passwordLegacyPolicy*** parameter.

For example:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -D "cn=directory manager" -w secret -p 389 -h
server.example.com -x
dn: cn=config
replace: passwordLegacyPolicy
passwordLegacyPolicy: off
```

14.5. CONFIGURING TIME-BASED ACCOUNT LOCKOUT POLICIES

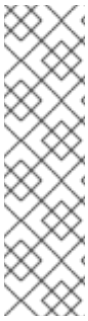
Aside from locking accounts for failed authentication attempts, another method of defining an account lockout policy is to base it on account inactivity or an account age. The Account Policy Plug-in uses a *relative* time setting to determine whether an account should be locked.

NOTE

Roles or classes of service can be used to inactivate accounts based on *absolute* account times. For example, a CoS can be created that inactivates every account created before a certain date.

The Account Policy Plug-in requires three configuration entries:

- A configuration entry for the plug-in itself. This sets global values that are used for all account policies configured on that server.
- An account policy configuration entry. This entry is within the user directory and is essentially a template which is referenced and applied to user account entries.
- An entry which applies the account policy entry. A user account can reference an account policy directly or a CoS or role can be used to apply account policies to sets of user accounts automatically.



NOTE

An account policy is applied through the ***acctPolicySubentry*** attribute. While this attribute can be added directly to user accounts, this attribute is single-valued – which means that only one account policy can be applied to that account.

That may be fine in most cases. However, an organization could realistically create two account policies, one for account inactivity and then another for account expiration based on age.

Using a CoS to apply account policies allows multiple account policies to be used for an account.

14.5.1. Account Policy Plug-in Syntax

The Account Policy Plug-in itself only has two configuration attributes:

- *nsslapd-pluginEnabled*, which sets whether the plug-in is enabled or disabled. This attribute is **off** by default.
- *nsslapd-pluginarg0*, which points to he DN of the plug-in configuration directory. The configuration entry is usually a child entry of the plug-in itself, such as **cn=config,cn=Account Policy Plugin,cn=plugins,cn=config**.

Past that, account policies are defined in two parts:

- The plug-in configuration entry identified in the *nsslapd-pluginarg0* attribute. This sets global configuration for the plug-in to use to identify account policy configuration entries and to manage user account entries. These settings apply across the server.

  The configuration entry attributes are listed in Table 14.5, “Account Policy Plug-in Attributes” .
- The account policy configuration entry. This is much like a template entry, which sets specific values for the account policies. User accounts – either directly or through CoS entries – reference this account policy entry.

The account policy and user entry attributes are listed in Table 14.6, "Account Policy Entry and User Entry Attributes".

Table 14.5. Account Policy Plug-in Attributes

Attribute	Definition
altstateattrname	Sets a fallback attribute for the plug-in to use to calculate the expiration time, if the ***stateattrname*** attribute does not exist in the user account.
alwaysRecordLogin	Sets whether to record the last login time for every user account, regardless of whether that account has an active account policy applied to it. By default, only entries with the ***acctPolicySubentry*** attribute on the entry have a login time recorded. Setting this to yes allows account policies to be applied indirectly, through roles or CoS.
limitattrname	Sets the attribute in the account policy which is used to evaluate the account status. For example, if the ***accountInactivityLimit*** attribute is used, then the account policy is evaluated based on how long the account has been inactive.
specattrname	Sets what attribute *on a user account* (or CoS or role) is used to flag that that entry has an account policy applied to it.
stateattrname	Sets the attribute for the plug-in to use to calculate the expiration time. For example, for a policy based on account inactivity, this is generally the last login time (***lastLoginTime***).

Table 14.6. Account Policy Entry and User Entry Attributes

Attribute	Definition	Configuration or User Entry
accountpolicy (object class)	Defines a template entry for account inactivation or expiration policies.	Configuration
accountInactivityLimit (attribute)	Sets the time period, in seconds, from the last login time of an account before that account is locked for inactivity.	Configuration
acctPolicySubentry (attribute)	Identifies any entry which belongs to an account policy (specifically, an account lockout policy). The value of this attribute points to the DN of the account policy which is applied to the entry.	User
createTimestamp (operational attribute)	Contains the date and time that the entry was initially created.	User
lastLoginTime (operational attribute)	Contains a timestamp of the last time that the given account authenticated to the directory.	User

14.5.2. Configuring Time-Based Account Lockout Policies

1. Enable the Account Policy Plug-in.

```
[user@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

```
dn: cn=Account Policy Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Set the *nsslapd-pluginarg0* attribute to point to the plug-in configuration entry.

```
[user@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Account Policy Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config
```

3. Create the plug-in configuration entry.

   - To use CoS or roles with account policies, set the ***alwaysRecordLogin*** value to **yes**. This means every entry has a login time recorded, even if it does not have the ***acctPolicySubentry*** attribute.
   - Set the primary attribute to use for the account policy evaluation as value for ***stateAttrName***. For account inactivity, use the **lastLoginTime** attribute. For a simple account expiration time, use **createTimestamp** attribute.
   - You can set a secondary attribute in ***altStateAttrName***, that is checked if the primary one defined in ***stateAttrName*** does not exist. If no attribute is specified as alternative the default value **createTimestamp** is used.

WARNING

If the value for the primary attribute is set to **lastLoginTime** and ***altStateAttrName*** to **createTimestamp**, users in existing environments are automatically locked out when their accounts do not have the **lastLoginTime** attribute and the ***createTimestamp*** is older than the configured inactivity period.

To avert this situation, set the alternative attribute to **1.1**. This explicitly states to use no attribute as alternative. The ***lastLoginTime*** attribute will be created automatically after the user logs in the next time. Alternatively you can write a script to add the **lastLoginTime** attribute to each user account.

   - Set the attribute to use to show which entries have an account policy applied to them (**acctPolicySubentry**).
   - Set the attribute in the account policy which is used to set the actual timeout period, in seconds (**accountInactivityLimit**).

```
[user@server ~]$ ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

objectClass: top
objectClass: extensibleObject
cn: config
alwaysRecordLogin: yes
stateAttrName: lastLoginTime
altStateAttrName: 1.1
specattrname: acctPolicySubentry
limitattrname: accountInactivityLimit
```

4. Restart the server to load the new plug-in configuration.

```
[user@server ~]$ service dirsrv start
```

5. Define an account policy.

```
[user@server ~]$ ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Account Inactivation Policy,dc=example,dc=com

objectClass: top
objectClass: ldapsubentry
objectClass: extensibleObject
objectClass: accountpolicy
accountInactivityLimit: 2592000
cn: Account Inactivation Policy
```

6. Create the class of service template entry.

```
[user@server ~]$ ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=TempltCoS,dc=example,dc=com

objectClass: top
objectClass: ldapsubentry
objectClass: extensibleObject
objectClass: cosTemplate
acctPolicySubentry: cn=Account Inactivation Policy,dc=example,dc=com
```

Account policies can be defined directly on user entries, instead of using a CoS. However, using a CoS allows an account policy to be applied and updated reliably for multiple entries and it allows multiple policies to be applied to an entry.

7. Create the class of service definition entry. The managed entry for the CoS is the account policy attribute, ***acctPolicySubentry***. This example applies the CoS to the entire directory tree.

```
[user@server ~]$ ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=DefnCoS,dc=example,dc=com

objectClass: top
objectClass: ldapsubentry
objectclass: cosSuperDefinition
objectclass: cosPointerDefinition
cosTemplateDn: cn=TempltCoS,dc=example,dc=com
cosAttribute: acctPolicySubentry default operational-default
```

14.5.3. Tracking Login Times without Setting Lockout Policies

It is also possible to use the Account Policy Plug-in to track user login times *without* setting an expiration time or inactivity period. In this case, the Account Policy Plug-in is used to add the ***lastLoginTime*** attribute to user entries, but no other policy rules need to be set.

In that case, set up the Account Policy Plug-in as normal, to track login times. However, do not create a CoS to act on the login information that is being tracked.

1. Enable the Account Policy Plug-in.

```
[user@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Account Policy Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. Set the *nsslapd-pluginarg0* attribute to point to the plug-in configuration entry.

```
[user@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Account Policy Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config
```

3. Create the plug-in configuration entry to record login times.

- Set the ***alwaysRecordLogin*** value to yes so that every entry has a login time recorded.
- Set the ***lastLoginTime*** attribute as the attribute to use for the account policy (***stateattrname***).
- Set the attribute to use to show which entries have an account policy applied to them (***acctPolicySubentry***).
- Set the attribute in the account policy which is used to set the actual timeout period, in seconds (***accountInactivityLimit***).

```
[user@server ~]$ ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

objectClass: top
objectClass: extensibleObject
cn: config
alwaysRecordLogin: yes
stateattrname: lastLoginTime
altstateattrname: createTimestamp
specattrname: acctPolicySubentry
limitattrname: accountInactivityLimit
```

4. Restart the server to load the new plug-in configuration.

```
[user@server ~]$ service dirsrv start
```

14.5.4. Unlocking Inactive Accounts

Accounts which are inactivated through the Account Policy Plug-in cannot be managed with the tools that are used to manage lockouts that are set manually by the administrator (**ns-activate.pl**) or through the password policy.

If an account is locked because it reached the inactivity limit, it can be reactivated by resetting the ***lastLoginTime*** attribute. For example:

```
[user@server ~]$ ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
changetype: modify
replace: lastLoginTime
lastLoginTime: 20160610080000Z
```

NOTE

The ***lastLoginTime*** is set in GMT/UTC time (Zulu time zone) indicated by the appended **Z** to the time stamp.

14.6. REPLICATING ACCOUNT LOCKOUT ATTRIBUTES

Account lockout policies will block a user ID from being able to access the Directory Server if the login attempt fails a set number of times. This prevents hackers or other malicious people from illegitimately accessing the Directory Server by guessing a password. Password policies are set locally, and generally account lockout attributes are local to each replica. This means that a person can attempt to log in to one replica until the account lockout count is reached, then try again immediately on another replica. The way to prevent that is to replicate the attributes related to the account lockout counts for an entry, so that the malicious user is locked out of every supplier and consumer replica in the configuration if a login attempt fails on a single master.

By default, three password policy attributes are not replicated, even if other password attributes are. These attributes are related to of login failures and lockout periods:

- ***passwordRetryCount***
- ***retryCountResetTime***
- ***accountUnlockTime***

14.6.1. Managing the Account Lockouts and Replication

Password and account lockout policies are enforced in a replicated environment slightly differently:

- Password policies are enforced on the data master.
- Account lockout is enforced on all servers participating in replication.

Some of the password policy information in the directory is replicated automatically:

- ***passwordMinAge*** and ***passwordMaxAge***
- ***passwordExp***
- ***passwordWarning***

However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications. Account lockout counters and tiers are not replicated, either, unless specifically configured for replication.

When configuring a password policy in a replicated environment, make sure that these elements are in place, so password policies and account lockout settings are enforced consistently:

- Warnings from the server of an impending password expiration are issued by all replicas. This information is kept locally on each server, so if a user binds to several replicas in turn, they will be issued the same warning several times. In addition, if the user changes the password, it may take time for this information to filter to the replicas. If a user changes a password and then immediately rebinds, he may find that the bind fails until the replica registers the changes.
- The same bind behavior should occur on all servers, including suppliers and replicas. Make sure to create the same password policy configuration information on each server.
- Account lockout counters may not work as expected in a multi-mastered environment. Account lockout counters are not replicated by default (although this can be configured). If account lockout attributes are not replicated at all, then a user could be locked out from one server but could successfully bind to another server (or, conversely, a user may be unlocked on one server and still blocked on another). If account lockout attributes are replicated, then there could be lags between an account lockout change on one server and when that change is propagated to the other servers. It depends on the replication schedule.
- Entries that are created for replication (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the ***passwordExpirationTime*** attribute to the entry, and give it a value of **20380119031407Z** (the top of the valid range).

NOTE

If the password policy is enabled and the ***alwaysRecordLogin*** parameter set to **yes**, the value of the ***lastLoginTime*** attribute can be different on masters and read-only replicas. For example, if a user logs in to a read-only replica, the ***lastLoginTime*** attribute is updated locally but the value is not replicated to the master servers.

14.6.2. Configuring Directory Server to Replicate Password Policy Attributes

A special core configuration attribute controls whether password policy operational attributes are replicated. This is the ***passwordIsGlobalPolicy*** attribute, which is enabled in the consumer Directory Server configuration to allow the consumer to accept password policy operational attributes.

By default, this attribute is set to **off**.

To enable these attributes to be replicated, change the ***passwordIsGlobalPolicy*** configuration attribute on the consumer:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x -h consumer1.example.com

dn: cn=config
changetype: modify
replace: passwordIsGlobalPolicy
passwordIsGlobalPolicy: on
```

Changing that value to **on** allows the ***passwordRetryCount***, ***retryCountResetTime***, and ***accountUnlockTime*** to be replicated. No other configuration is necessary for the attributes to be included with the replicated attributes.

14.6.3. Configuring Fractional Replication for Password Policy Attributes

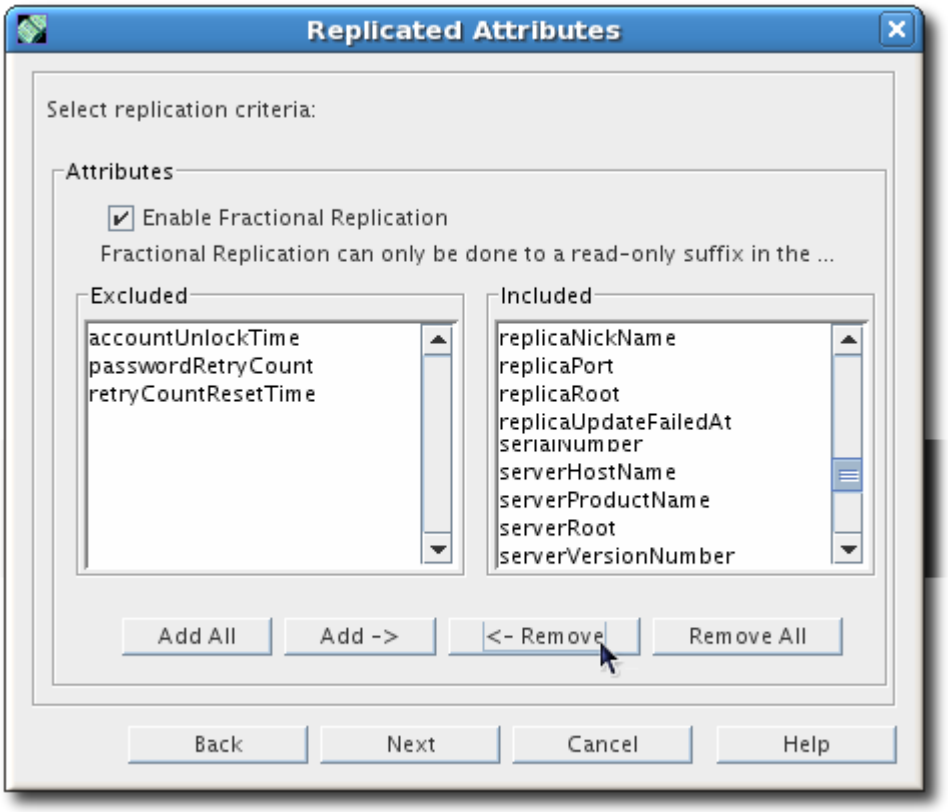
Setting the ***passwordIsGlobalPolicy*** attribute affects the consumer in replication, in that it allows the consumer to receive updates to those attributes. To control whether the password policy attributes are actually replicated by the supplier, use fractional replication, which controls what specific entry attributes are replicated.

If the password policy attributes should be replicated, then make sure these attributes are included in the fractional replication agreement (as they are by default).

If the ***passwordIsGlobalPolicy*** attribute is set to **off** on the consumer, so no password policy attributes should be replicated, use fractional replication (described in Section 11.1.7, "Replicating a Subset of Attributes with Fractional Replication") to enforce that on the supplier and specifically exclude those attributes from the replication agreement.

1. When configuring the replication agreement on the supplier, as described (for example) in Section 11.4.3, "Creating the Replication Agreement", select the **Enable Fractional Replication** check box.

2. By default, every attribute is listed in the **Replicated Attributes** box. Select the ***passwordRetryCount***, ***retryCountResetTime***, and ***accountUnlockTime*** parameters and click the arrow button to move them into the **Do Not Replicate** box.



3. Finish configuring the replication agreement.

14.7. SYNCHRONIZING PASSWORDS

Password changes in a Directory Server entry can be synchronized to password attributes in Active Directory entries by using the **Password Sync** utility.

When passwords are synchronized, password policies are enforced on each sync peer locally. The syntax or minimum length requirements on the Directory Server apply when the password is changed in the Directory Server. When the changed password is synced over to the Windows server, the Windows password policy is enforced. The password policies themselves are not synchronized.

Configuration information is kept locally and cannot be synchronized, including the password change history and the account lockout counters.

When configuring a password policy for synchronization, consider the following points:

- The **Password Sync** utility must be installed locally on the Windows machine that will be synchronized with a Directory Server.

- **Password Sync** can only link the Windows machine to a single Directory Server; to sync changes with multiple Directory Server instances, configure the Directory Server for multi-master replication.
- Password expiration warnings and times, failed bind attempts, and other password-related information is enforced locally per server and is not synchronized between sync peer servers.
- The same bind behavior should occur on all servers. Make sure to create the same or similar password policies on both Directory Server and Active Directory servers.
- Entries that are created for synchronization (for example, the server identities) need to have passwords that never expire. To make sure that these special users have passwords that do not expire, add the ***passwordExpirationTime*** attribute to the Directory Server entry, and give it a value of **20380119031407Z** (the top of the valid range).

See Chapter 12, *Synchronizing Red Hat Directory Server with Microsoft Active Directory* for more information on synchronizing Directory Server and Windows users and passwords.

14.8. ENABLING DIFFERENT TYPES OF BINDS

Whenever an entity logs into or accesses the Directory Server, it *binds* to the directory. There are many different types of bind operation, sometimes depending on the method of binding (such as simple binds or autobind) and some depending on the identity of user binding to the directory (anonymous and unauthenticated binds).

The following sections contain configuration parameters that can increase the security of binds (as in Section 14.8.1, "Requiring Secure Binds") or streamline bind operations (such as Section 14.8.4, "Configuring Autobind").

14.8.1. Requiring Secure Binds

A simple bind is when an entity uses a simple bind DN-password combination to authenticate to the Directory Server. Although it is possible to use a password file rather than sending a password directly through the command line, both methods still require sending or accessing a plaintext password over the wire. That makes the password vulnerable to anyone sniffing the connection.

It is possible to require simple binds to occur over a secure connection (SSL/TLS or Start TLS), which effectively encrypts the plaintext password as it is sent with the bind operation. (It is also possible to use alternatives to simple binds, such as SASL authentication and certificate-based authentication.)

IMPORTANT

Along with regular users logging into the server and LDAP operations, server-to-server connections are affected by requiring secure connections for simple binds. Replication, synchronization, and database chaining can all use simple binds between servers, for instance.

Make sure that replication and sync agreements and chaining configuration specify secure connections if the ***nsslapd-require-secure-binds*** attribute is turned on. Otherwise, these operations will fail.

NOTE

Requiring a secure connection for bind operations only applies to *authenticated binds*. Bind operations without a password (anonymous and unauthenticated binds) can proceed over standard connections.

1. Add the ***nsslapd-require-secure-binds*** attribute to the **cn=config** entry:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-require-secure-binds
nsslapd-require-secure-binds: on
```

2. Restart the server.

```
service dirsrv restart
```

14.8.2. Disabling Anonymous Binds

If a user attempts to connect to the Directory Server without supplying any user name or password, this is an *anonymous bind*. Anonymous binds simplify common search and read operations, like checking the directory for a phone number or email address, by not requiring users to authenticate to the directory first.

NOTE

By default, anonymous binds are allowed (on) for search and read operations. This allows access to *regular directory entries*, which includes user and group entries as well as configuration entries like the root DSE. A different option, **rootdse**, allows anonymous search and read access to search the root DSE itself, but restricts access to all other directory entries.

However, there are risks with anonymous binds. Adequate ACIs must be in place to restrict access to sensitive information and to disallow actions like modifies and deletes. Additionally, anonymous binds can be used for denial of service attacks or for malicious people to gain access to the server.

Section 13.9.1, “Granting Anonymous Access” has examples on setting ACIs to control what anonymous users can access, and Section 10.1.5, “Setting Resource Limits on Anonymous Binds” has information on placing resource limits for anonymous users.

If those options do not offer a sufficient level of security, then anonymous binds can be disabled entirely.

1. Add the ***nsslapd-allow-anonymous-access*** attribute to the **cn=config** entry:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-allow-anonymous-access
nsslapd-allow-anonymous-access: off
```

2. Restart the server.

```
service dirsrv restart
```

NOTE

When anonymous binds are disabled, unauthenticated binds are also disabled automatically.

14.8.3. Allowing Unauthenticated Binds

An *unauthenticated bind* is a bind where the user supplies a user name but not a password. For example, running an **ldapsearch** without supplying a password option:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x
"(objectclass=*)"
```

When unauthenticated binds are allowed, the bind attempt goes through as an anonymous bind.

Unauthenticated binds are less secure than authenticated binds, and in some directories can be used to circumvent ACIs or performs denial-of-service attacks. This is why in Directory Server unauthenticated binds are disabled by default. If a user tries to bind without a password, the attempt fails:

```
ldap_simple_bind: DSA is unwilling to perform
ldap_simple_bind: additional info: Unauthenticated binds are not allowed
```

Unauthenticated binds only apply to bind attempts where a password is not given but a bind identity is. If the wrong password is given, the operation fails with an invalid credentials error:

```
ldap_simple_bind: Invalid credentials
```

If no bind ID or password is given, then the directory returns whatever information is allowed for an anonymous bind.

The ***nsslapd-allow-unauthenticated-binds*** attribute sets whether to allow an unauthenticated bind to succeed as an anonymous bind. Setting this parameter to **on** allows unauthenticated binds. By default, this parameter is **off**.

1. To configure unauthenticated binds, edit the Directory Server **cn=config** entry:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
```

```
dn: cn=config
changetype:replace
replace: nsslapd-allow-unauthenticated-binds
nsslapd-allow-unauthenticated-binds: on
```

2. Restart the server.

```
service dirsrv restart
```

14.8.4. Configuring Autobind

Autobind is a way to connect to the Directory Server based on local UNIX credentials, which are mapped to an identity stored in the directory itself. Autobind is configured in two parts:

Before configuring autobind, first make sure that LDAPI is enabled (in Section 1.5, "Enabling LDAPI"). Then, configure the autobind mappings (in Section 14.8.4.2, "Configuring Autobind").

14.8.4.1. Overview of Autobind and LDAPI

Inter-process communication (IPC) is a way for separate processes on a Unix machine or a network to communicate directly with each other. *LDAPI* is a way to run LDAP connections over these IPC connections, meaning that LDAP operations can run over Unix sockets. These connections are much faster and more secure than regular LDAP connections.

The Directory Server uses these LDAPI connections to allow users to bind immediately to the Directory Server or to access the Directory Server using tools which support connections over Unix sockets. Autobind uses the *uid:gid* of the Unix user and maps that user to an entry in the Directory Server, then allows access for that user.

Autobind allows mappings to three directory entries:

- User entries, if the Unix user matches one user entry
- Directory Manager (or the super user defined in ***nsslapd-ldapimaprootdn***), if the Unix user is **root**

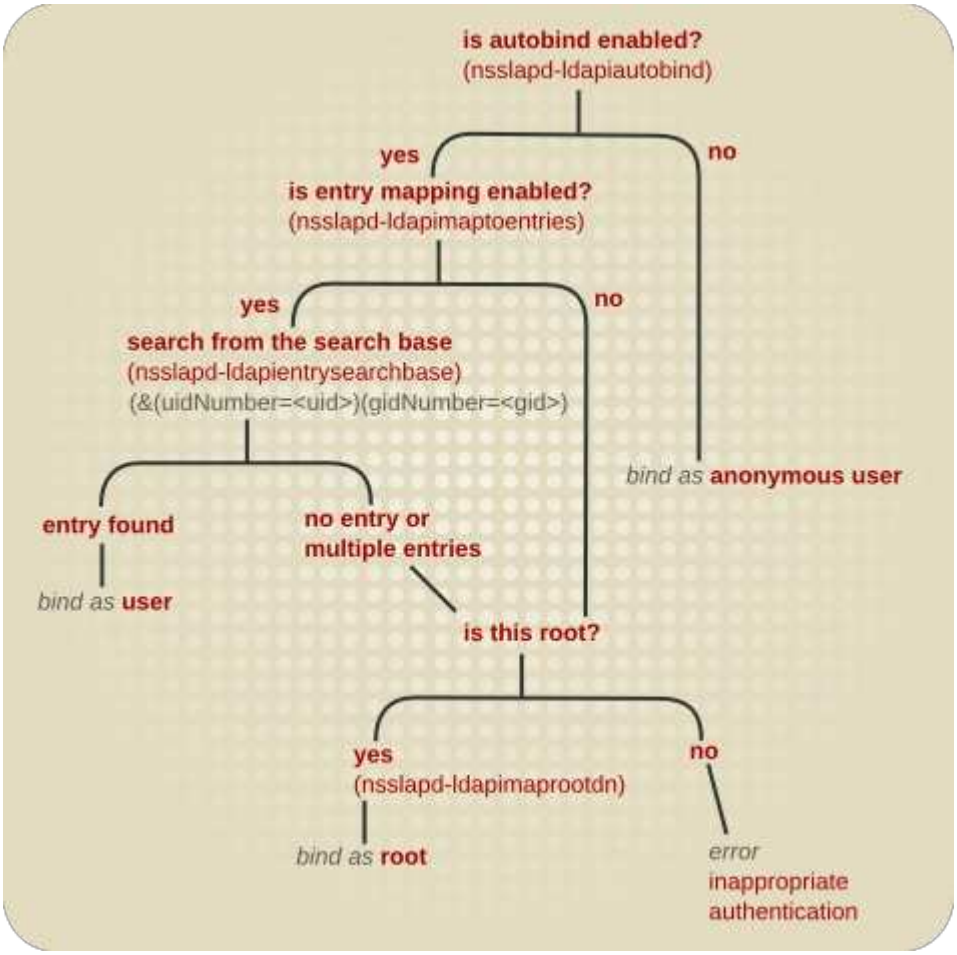


Figure 14.1. Autobind Connection Path

The special autobind users are entries beneath a special autobind suffix (outside the regular user subtree). The entries underneath are identified by their user and group ID numbers:

```
gidNumber=gid+uidNumberuid, autobindsuffix
```

If autobind is not enabled but LDAPI is, then Unix users are anonymously bound to the Directory Server, unless they provide other bind credentials.

NOTE

Autobind allows a client to send a request to the Directory Server without supplying a bind user name and password or using other SASL authentication mechanism. According to the LDAP standard, if bind information is not given with the request, the server processes the request as an anonymous bind. To be compliant with the standard, which requires some kind of bind information, any client that uses autobind should send the request with SASL/EXTERNAL.

For more information on configuring SASL, see Section 7.11, “Setting up SASL Identity Mapping” .

14.8.4.2. Configuring Autobind

Configuring autobind alone allows anonymous access to the Directory Server. It is possible to enable mapping Unix users to entries and also to map **root** to Directory Manager.

1. Run **ldapmodify** to update the Directory Server configuration.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
```

2. Enable autobind.

```
replace: nsslapd-ldapiautobind
nsslapd-ldapiautobind: on
```

3. To map user entries, add four attributes:
   - ***nsslapd-ldapimaptoentries*** to enable entry mapping
   - ***nsslapd-ldapiuidnumbertype*** to set the Directory Server attribute to map to the Unix UID number
   - ***nsslapd-ldapigidnumbertype*** to set the Directory Server attribute to map to the Unix group ID number
   - ***nsslapd-ldapientrysearchbase*** to set the search base to use to find Directory Server user entries

```
add: nsslapd-ldapimaptoentries
nsslapd-ldapimaptoentries: on
-
add: nsslapd-ldapiuidnumbertype
nsslapd-ldapiuidnumbertype: uidNumber
-
add: nsslapd-ldapigidnumbertype
nsslapd-ldapigidnumbertype: gidNumber
-
add: nsslapd-ldapientrysearchbase
nsslapd-ldapientrysearchbase: ou=people,dc=example,dc=com
```

4. To map the **root** entry to Directory Manager, add the ***nsslapd-ldapimaprootdn*** attribute:

```
add: nsslapd-ldapimaprootdn
nsslapd-ldapimaprootdn: cn=Directory Manager
```

5. Restart the server to apply the new configuration.

```
service dirsrv restart example
```

14.9. USING PASS-THROUGH AUTHENTICATION

Pass-through authentication (PTA) is a mechanism which allows one Red Hat Directory Server instance to consult another to authenticate bind requests. Pass-through authentication is implement through the PTA Plug-in; when enabled, the plug-in lets a Directory Server instance accept simple bind operations (password-based) for entries not stored in its local database.

Directory Server uses PTA to administer the user and configuration directories on separate instances of Directory Server.

If the configuration directory and the user directory are installed on separate instances of Directory Server, the setup program automatically sets up PTA to allow the Configuration Administrator user (usually **admin**) to perform administrative duties.

PTA is required in this case because the **admin** user entry is stored under **o=NetscapeRoot** suffix in the configuration directory. Therefore, attempts to bind to the user directory as **admin** would normally fail. PTA allows the user directory to transmit the credentials to the configuration directory, which verifies them. The user directory then allows the **admin** user to bind.

The user directory in this example acts as the *PTA Directory Server*, the server that passes through bind requests to another Directory Server. The configuration directory acts as the *authenticating directory*, the server that contains the entry and verifies the bind credentials of the requesting client.

The *pass-through subtree* is the subtree *not* present on the PTA directory. When a user's bind DN contains this subtree, the user's credentials are passed on to the authenticating directory.

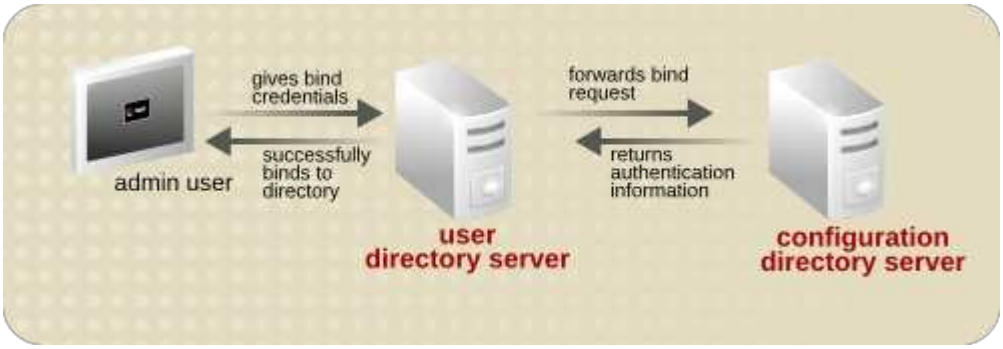


Figure 14.2. Simple Pass-Through Authentication Process

NOTE

The PTA Plug-in may not be listed in the Directory Server Console if the same server instance is used for the user directory and the configuration directory.

Here's how pass-through authentication works:

1. The configuration Directory Server (authenticating directory) is installed on machine A. The configuration directory always contains the configuration database and suffix, **o=NetscapeRoot**. In this example, the server name is **configdir.example.com**.

2. The user Directory Server (PTA directory) is then installed on machine B. The user directory stores the root suffix, such as **dc=example,dc=com**. In this example, the server name is **userdir.example.com**.

3. When the user directory is set up on machine B, the setup script prompts for the LDAP URL of the configuration directory on machine A.

4. The setup program enables the PTA Plug-in and configures it to use the configuration directory LDAP URL.

   This entry contains the LDAP URL for the configuration directory. For example:

   ```
   dn: cn=Pass Through Authentication,cn=plugins,
   ...
   nsslapd-pluginEnabled: on
   nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
   ...
   ```

   The user directory is now configured to send all bind requests for entries with a DN containing **o=NetscapeRoot** to the configuration directory **configdir.example.com**.

5. When installation is complete, the **admin** user attempts to connect to the user directory to begin adding users.

6. The setup program adds the **admin** user's entry to the directory as **uid=admin,ou=TopologyManagement,o=NetscapeRoot**. So the user directory passes the bind request through to the configuration directory as defined by the PTA Plug-in configuration.

7. The configuration directory authenticates the user's credentials and sends the information back to the user directory.

8. The user directory allows the **admin** user to bind.

14.9.1. PTA Plug-in Syntax

PTA Plug-in configuration information is specified in the **cn=Pass Through Authentication**, **cn=plugins,cn=config** entry on the PTA directory (the user directory configured to pass through bind requests to the authenticating directory) using the required PTA syntax. There are only two attributes in this entry that are significant:

- *nsslapd-pluginEnabled*, which sets whether the plug-in is enabled or disabled. The value for this attribute can be **on** or **off**.

- *nsslapd-pluginarg0*, which points to the configuration directory. The value for this attribute is the LDAP URL of the server and suffix to which to pass the bind requests, along with the optional parameters, *maxconns*, *maxops*, *timeout*, *ldver*, *connlifetime*, *startTLS*.

The variable components of the PTA plug-in syntax are described in Table 14.7, "PTA Plug-in Parameters".

NOTE

The LDAP URL (**ldap|ldaps:**//*authDS/subtree*) must be separated from the optional parameters (*maxconns, maxops, timeout, ldver, connlifetime, startTLS*) by a single space. If any of the optional parameters are defined, all of them must be defined, even if only the default values are used.

Several authenticating directories or subtrees can be specified by incrementing the ***nsslapd-pluginarg*** attribute suffix by one each time, as in Section 14.9.3.2, "Specifying Multiple Authenticating Directory Servers". For example:

```
nsslapd-pluginarg0: LDAP URL for the first server
nsslapd-pluginarg1: LDAP URL for the second server
nsslapd-pluginarg2: LDAP URL for the third server
...
```

The optional parameters are described in the following table in the order in which they appear in the syntax.

Table 14.7. PTA Plug-in Parameters

Variable	Definition
state	Defines whether the plug-in is enabled or disabled. Acceptable values are **on** or **off**.
ldap|ldaps	Defines whether SSL is used for communication between the two Directory Servers. See Section 14.9.2.1, "Configuring the Servers to Use a Secure Connection" for more information.
authDS	The authenticating directory host name. The port number of the Directory Server can be given by adding a colon and then the port number. For example, **ldap://dirserver.example.com:389**/. If the port number is not specified, the PTA server attempts to connect using either of the standard ports: Port 389 if **ldap://** is specified in the URL. Port 636 if **ldaps://** is specified in the URL. See Section 14.9.2.2, "Specifying the Authenticating Directory Server" for more information.

Variable	Definition
subtree	The *pass-through subtree*. The PTA Directory Server passes through bind requests to the authenticating Directory Server from all clients whose DN is in this subtree. See Section 14.9.2.3, "Specifying the Pass-Through Subtree" for more information. This subtree must not exist on this server. To pass the bind requests for **o=NetscapeRoot** to the configuration directory, the subtree **o=NetscapeRoot** must not exist on the server.
maxconns	*Optional*. The maximum number of connections the PTA directory can simultaneously open to the authenticating directory. The default is **3**. See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.
maxops	*Optional*. The maximum number of simultaneous operations (usually bind requests) the PTA directory can send to the authenticating directory within a single connection. The default is **5**. See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.
timeout	*Optional*. The time limit, in seconds, that the PTA directory waits for a response from the authenticating Directory Server. If this timeout is exceeded, the server returns an error to the client. The default is **300** seconds (five minutes). Specify zero (**0**) to indicate no time limit should be enforced. See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.
ldver	*Optional*. The version of the LDAP protocol used to connect to the authenticating directory. Directory Server supports LDAP version 2 and 3. The default is version 3, and Red Hat strongly recommends *against* using LDAPv2, which is old and will be deprecated. See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.
connlifetime	*Optional*. The time limit, in seconds, within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating directory. The server will not close the connection unless a bind request is initiated and the directory determines the connection lifetime has been exceeded. If this option is not specified, or if only one host is listed, no connection lifetime will be enforced. If two or more hosts are listed, the default is **300** seconds (five minutes). See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.

Variable	Definition
startTLS	*Optional.* A flag of whether to use Start TLS for the connection to the authenticating directory. Start TLS establishes a secure connection over the standard port, so it is useful for connecting using LDAP instead of LDAPS. The SSL server and CA certificates need to be available on both of the servers. The default is **0**, which is off. To enable Start TLS, set it to**1**. To use Start TLS, the LDAP URL must use **ldap:**, not **ldaps:**. See Section 14.9.2.4, "Configuring the Optional Parameters" for more information.

14.9.2. Configuring the PTA Plug-in

The only method for configuring the PTA plug-in is to modify the entry **cn=Pass Through Authentication,cn=plugins,cn=config**. To modify the PTA configuration:

1. Use the **ldapmodify** command to modify **cn=Pass Through Authentication,cn=plugins,cn=config.**
2. Restart Directory Server.

Before configuring any of the PTA Plug-in parameters, the PTA Plug-in entry must be present in the Directory Server. If this entry does not exist, create it with the appropriate syntax, as described in Section 14.9.1, "PTA Plug-in Syntax".

NOTE

If the user and configuration directories are installed on different instances of the directory, the PTA Plug-in entry is automatically added to the user directory's configuration and enabled.

This section provides information about configuring the plug-in in the following sections:

- Section 14.9.2.1, "Configuring the Servers to Use a Secure Connection"
- Section 14.9.2.2, "Specifying the Authenticating Directory Server"
- Section 14.9.2.3, "Specifying the Pass-Through Subtree"
- Section 14.9.2.4, "Configuring the Optional Parameters"

14.9.2.1. Configuring the Servers to Use a Secure Connection

The PTA directory can be configured to communicate with the authenticating directory over SSL by specifying LDAPS in the LDAP URL of the PTA directory. For example:

```
nsslapd-pluginarg0: ldaps://ldap.example.com:636/o=NetscapeRoot
```

14.9.2.2. Specifying the Authenticating Directory Server

The authenticating directory contains the bind credentials for the entry with which the client is attempting to bind. The PTA directory passes the bind request to the host defines as the authenticating directory. To specify the authenticating Directory Server, replace *authDS* in the LDAP URL of the PTA directory with the authenticating directory's host name, as described in Table 14.7, "PTA Plug-in Parameters".

1. Use **ldapmodify** edit the PTA Plug-in entry.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot
```

Optionally, include the port number. If the port number is not given, the PTA Directory Server attempts to connect using either the standard port (389) for **ldap:**// or the secure port (636) for **ldaps:**//.

If the connection between the PTA Directory Server and the authenticating Directory Server is broken or the connection cannot be opened, the PTA Directory Server sends the request to the next server specified, if any. There can be multiple authenticating Directory Servers specified, as required, to provide failover if the first Directory Server is unavailable. All of the authentication Directory Server are set in the ***nsslapd-pluginarg0*** attribute.

Multiple authenticating Directory Servers are listed in a space-separate list of *host:port* pairs, with this format:

```
ldap|ldaps://host1:port1 host2:port2/subtree
```

2. Restart the server.

```
service dirsrv restart instance_name
```

14.9.2.3. Specifying the Pass-Through Subtree

The PTA directory passes through bind requests to the authenticating directory from all clients with a DN defined in the pass-through subtree. The subtree is specified by replacing the *subtree* parameter in the LDAP URL of the PTA directory.

The pass-through subtree must not exist in the PTA directory. If it does, the PTA directory attempts to resolve bind requests using its own directory contents and the binds fail.

1. Use the **ldapmodify** command to import the LDIF file into the directory.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot
```

For information on the variable components in this syntax, see Table 14.7, “PTA Plug-in Parameters”.

2. Restart the server.

```
service dirsrv restart instance_name
```

14.9.2.4. Configuring the Optional Parameters

Additional parameters the control the PTA connection can be set with the LDAP URL.

```
ldap|ldaps://authDS/subtree maxconns, maxops, timeout, ldver, connlifetime, startTLS
```

- The maximum number of connections the PTA Directory Server can open simultaneously to the authenticating directory, represented by *maxconns* in the PTA syntax. The default value is **3**.
- The maximum number of bind requests the PTA Directory Server can send simultaneously to the authenticating Directory Server within a single connection. In the PTA syntax, this parameter is *maxops*. The default is value is **5**.
- The time limit for the PTA Directory Server to wait for a response from the authenticating Directory Server. In the PTA syntax, this parameter is *timeout*. The default value is **300** seconds (five minutes).
- The version of the LDAP protocol for the PTA Directory Server to use to connect to the authenticating Directory Server. In the PTA syntax, this parameter is *ldver*. The default is **LDAPv3**.
- The time limit in seconds within which a connection may be used. If a bind request is initiated by a client after this time has expired, the server closes the connection and opens a new connection to the authenticating Directory Server. The server will not close the connection unless a bind request is initiated and the server determines the timeout has been exceeded. If this option is not specified or if only one authenticating Directory Server is listed in the *authDS* parameter, no time limit will be enforced. If two or more hosts are listed, the default is **300** seconds (five minutes). In the PTA syntax, this parameter is *connlifetime*.
- Whether to use Start TLS for the connection. Start TLS creates a secure connection over a standard LDAP port. For Start TLS, the servers must have their server and CA certificates installed, but they do not need to be running in SSL.

The default is **0**, which means Start TLS is off. To enable Start TLS, set it to **1**. To use Start TLS, the LDAP URL must use **ldap:**, not **ldaps:**.

1. Use **ldapmodify** to edit the plug-in entry.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn:  cn=Pass Through Authentication,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginarg0
nsslapd-pluginarg0: ldap://dirserver.example.com/o=NetscapeRoot 3,5,300,3,300,0
```

(In this example, each of the optional parameters is set to its default value.) Make sure there is a space between the *subtree* parameter, and the optional parameters.

NOTE

Although these parameters are optional, if any one of them is defined, they all must be defined, even if they use the default values.

2. Restart the server.

```
service dirsrv restart instance_name
```

14.9.3. PTA Plug-in Syntax Examples

This section contains the following examples of PTA Plug-in syntax in the **dse.ldif** file:

- Section 14.9.3.1, "Specifying One Authenticating Directory Server and One Subtree"
- Section 14.9.3.2, "Specifying Multiple Authenticating Directory Servers"
- Section 14.9.3.3, "Specifying One Authenticating Directory Server and Multiple Subtrees"
- Section 14.9.3.4, "Using Non-Default Parameter Values"
- Section 14.9.3.5, "Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers"

14.9.3.1. Specifying One Authenticating Directory Server and One Subtree

This example configures the PTA Plug-in to accept all defaults for the optional variables. This configuration causes the PTA Directory Server to connect to the authenticating Directory Server for all bind requests to the **o=NetscapeRoot** subtree. The host name of the authenticating Directory Server is **configdir.example.com**.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
...
```

14.9.3.2. Specifying Multiple Authenticating Directory Servers

If the connection between the PTA Directory Server and the authenticating Directory Server is broken or the connection cannot be opened, the PTA Directory Server sends the request to the next server specified, if any. There can be multiple authenticating Directory Servers specified, as required, to provide failover if the first Directory Server is unavailable. All of the authentication Directory Server are set in the ***nsslapd-pluginarg0*** attribute. Multiple authenticating Directory Servers are listed in a space-separate list of *host:port* pairs. For example:

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com:389 config2dir.example.com:1389/o=NetscapeRoot
...
```

NOTE

The ***nsslapd-pluginarg0*** attribute sets the authentication Directory Server; additional ***nsslapd-pluginargN*** attributes can set additional *suffixes* for the PTA Plug-in to use, but not additional *hosts*.

14.9.3.3. Specifying One Authenticating Directory Server and Multiple Subtrees

The following example configures the PTA Directory Server to pass through bind requests for more than one subtree (using parameter defaults):

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
nsslapd-pluginarg1: ldap://configdir.example.com/dc=example,dc=com
...
```

14.9.3.4. Using Non-Default Parameter Values

This example uses a non-default value (**10**) only for the maximum number of connections parameter **maxconns**. Each of the other parameters is set to its default value. However, because one parameter is specified, all parameters must be defined explicitly in the syntax.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot 10,5,300,3,300,1
...
```

14.9.3.5. Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers

To specify a different pass-through subtree and optional parameter values for each authenticating Directory Server, set more than one LDAP URL/optional parameters pair. Separate the LDAP URL/optional parameter pairs with a single space as follows.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0:ldap://configdir.example.com/o=NetscapeRoot 10,15,30,3,600,0
nsslapd-pluginarg1:ldap://config2dir.example.com/dc=example,dc=com 7,7,300,3,300,1
...
```

14.10. USING PAM FOR PASS-THROUGH AUTHENTICATION

Pass-through authentication is when any authentication request is forwarded from one server to another service.

Many systems already have authentication mechanisms in place for Unix and Linux users. One of the most common authentication frameworks is *Pluggable Authentication Modules* (PAM). Since many networks already existing authentication services available, administrators may want to continue using those services. A PAM module can be configured to tell Directory Server to use an existing authentication store for LDAP clients.

PAM pass-through authentication in Red Hat Directory Server uses the PAM Pass-Through Authentication Plug-in, which enables the Directory Server to talk to the PAM service to authenticate LDAP clients.

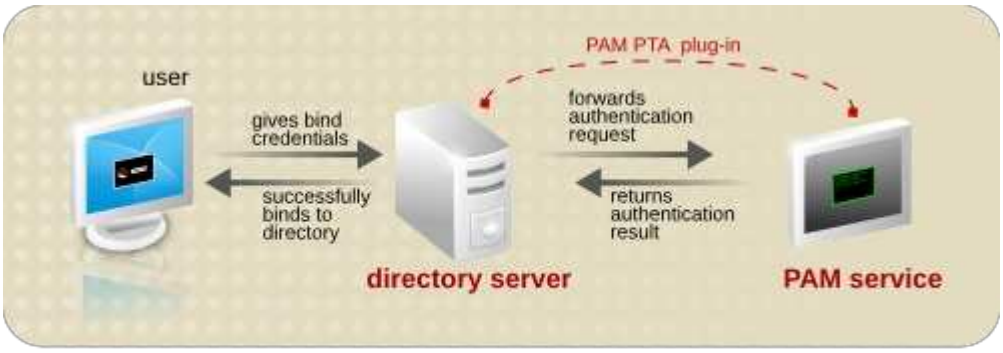


Figure 14.3. PAM Pass-Through Authentication Process

NOTE

PAM pass-through authentication works together with account inactivation when authenticating users, assuming that the appropriate mapping method (ENTRY) is used. However, PAM pass-through authentication does not validate passwords against password policies set either globally or locally, because the passwords are set and stored in the PAM module, not in the Directory Server.

14.10.1. PAM Pass-Through Authentication Configuration Options

PAM pass-through authentication is configured in child entries beneath the PAM Pass-Through Authentication plug-in container entry. There can be multiple PAM pass-through authentication policies, applied to different suffixes or to different entries within suffixes.

There are several different areas that can be configured for PAM pass-through:

- The suffixes that are controlled by the PAM pass-through authentication plug-in. This covers suffixes to exclude, suffixes to include, and how to handle a missing suffix.
- Individual entries within the configured suffixes which are the target of the authentication configuration. By default, all entries within a suffix are included in the authentication scope, but it is possible to configure multiple, different PAM Pass-Through Auth plug-in instances and then apply different plug-in configuration to different users.
- The PAM attribute mapping. The credentials that are offered to the Directory Server have to be mapped in some way to an LDAP entry and then, back to the credentials in the PAM service. This is done by defining a mapping method and then, optionally, which LDAP attribute to use to match the credentials.
- General configuration such as using SSL connections, the PAM service to use, and whether to fallback to LDAP authentication if PAM authentication fails.

NOTE

There can be multiple configuration instances of the PAM Pass-Through Authentication plug-in. An instance of the PAM Pass-Through Authentication plug-in can be applied to a subset of user entries by using the ***pamFilter*** attribute to set an LDAP filter to search for the specific entries to use with the plug-in.

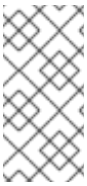
Table 14.8. PAM Pass-Through Auth Plug-in Attributes

Attribute	Definition
pamExcludeSuffix	Identifies suffixes to exclude from PAM authentication.
pamIncludeSuffix	Identifies suffixes to include for PAM authentication.
pamMissingSuffix	Identifies how to handle missing include or exclude suffixes. The options are ERROR (which causes the bind operation to fail); ALLOW, which logs an error but allows the operation to proceed; and IGNORE, which allows the operation and does not log any errors.

Attribute	Definition
pamFilter	Sets an LDAP filter to use to identify specific entries within the included suffixes for which to use PAM pass-through authentication. If not set, all entries within the suffix are targeted by the configuration entry.
pamIDAttr	Sets the name of the attribute holding the PAM ID.
pamIDMapMethod	Gives the method to use to map the LDAP bind DN to a PAM identity.  **NOTE** Directory Server user account inactivation is only validated using the ENTRY mapping method. With RDN or DN, a Directory Server user whose account is inactivated can still bind to the server successfully.
pamFallback	Sets whether to fallback to regular LDAP authentication if PAM authentication fails.
pamSecure	Requires secure (TLS/SSL) connection for PAM authentication.
pamService	Contains the service name to pass to PAM. This assumes that the service specified has a configuration file in **/etc/pam.d**.  **IMPORTANT** The **pam_fprintd.so** module cannot be in the configuration file referenced by the ***pamService*** attribute of the PAM Pass-Through Authentication Plug-in configuration. Using the PAM **fprintd** module causes the Directory Server to hit the max file descriptor limit and can cause the Directory Server process to abort.
nsslapd-pluginConfigArea	Specifies a different container entry to use as the parent for PAM pass-through authentication child entries. By default, the PAM Pass-Through Auth Plug-in entry is used as the parent entry for the configuration entries. This is set in the PAM Pass-Through Auth Plug-in entry. If a different container entry is used, then all PAM pass-through authentication child entries must be located beneath that container entry. All child entries in the specified location must belong to the **pamConfig** object class, but neither the container entry nor the PAM Pass-Through Auth Plug-in entry must belong to the **pamConfig** object class in that case.

14.10.1.1. Specifying the Suffixes to Target for PAM PTA

The PAM PTA plug-in is applied globally, to all suffixes, by default unless they are explicitly excluded. Excluding and including suffixes can help target what areas in the directory use PAM authentication instead of LDAP authentication.



NOTE

The target of a PAM pass-through authentication entry must be a suffix, not an arbitrary subtree. As described in Section 2.1, "Creating and Maintaining Suffixes", a suffix is a subtree which is associated with a specific back end database, such as **cn=config** which is associated with **NetscapeRoot** or the root suffix **dc=example,dc=com** which is associated with **userRoot**.

The ***pamExcludeSuffix*** attribute excludes a suffix. By default, only the configuration subtree (**cn=config**) is excluded. Alternatively, the PAM PTA plug-in can be applied to a suffix with the ***pamIncludeSuffix*** attribute. Both of these attributes are multi-valued.

If the include attribute is set, for example, all other suffixes are automatically excluded. Likewise, if an exclude attribute is set, all other suffixes are automatically included.

```
pamExcludeSuffix: cn=config
pamExcludeSuffix: o=NetscapeRoot
```

With ***pamIncludeSuffix***, only the given suffix is included and all others are automatically excluded. Since this attribute is multi-valued, more than one suffix can be included in the PAM evaluation by explicitly listing the suffixes.

```
pamIncludeSuffix: ou=Engineering,dc=example,dc=com
pamIncludeSuffix: ou=QE,dc=example,dc=com
```

The ***pamMissingSuffix*** attribute tells the server how to handle a failure if the specified suffix (include or exclude) does not exist. If it is set to **IGNORE**, then if the suffix does not exist, the plug-in simply skips that suffix and tries the next.

```
pamMissingSuffix: IGNORE
pamIncludeSuffix: ou=Engineering,dc=example,dc=com
pamIncludeSuffix: ou=Not Real,dc=example,dc=com
```

14.10.1.2. Applying Different PAM Pass-Through Authentication Configurations to Different Entries

By default, a PAM pass-through authentication policy applies to all entries within the designated suffixes. However, it is possible to specify an LDAP filter in the ***pamFilter*** attribute which identifies specific entries within the suffix to which to apply the PAM pass-through authentication policy.

This is useful for applying different PAM configurations or mapping methods to different user types, using multiple PAM pass-through authentication policies.

14.10.1.3. Setting PAM PTA Mappings

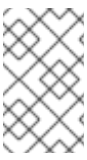
There has to be a way to connect the LDAP identity to the PAM identity. The first thing to define is the *method* to use to map the entries. There are three options: DN, RDN, and ENTRY. ENTRY uses a user-defined attribute in the entry.

Multiple mapping methods can be supplied in an ordered, space-separated list. The plug-in attempts to use each mapping method in the order listed until authentication succeeds or until it reaches the end of the list.

For example, this mapping method first maps the RDN method, then ENTRY, then DN, in the order the methods are listed:

```
pamIDMapMethod: RDN ENTRY DN
```

The different mapping methods are listed in Table 14.9, "Mapping Methods for PAM Authentication" .



NOTE

Directory Server user account inactivation is only validated using the ENTRY mapping method. With RDN or DN, a Directory Server user whose account is inactivated can still bind to the server successfully.

Table 14.9. Mapping Methods for PAM Authentication

Mapping	Description

Mapping	Description
RDN	This method uses the value from the leftmost RDN in the bind DN. The mapping for this method is defined by Directory Server. This is the default mapping method, if none is given.
ENTRY	This method pulls the value of the PAM identity from a user-defined attribute in the bind DN entry. The identity attribute is defined in the ***pamIDAttr*** attribute. `pamIDAttr: customPamUid`
DN	This method uses the full distinguished name from the bind DN. The mapping for this method is defined by Directory Server.

14.10.1.4. Configuring General PAM PTA Settings

Three general configuration settings can be set for PAM authentication:

- The service name to send to PAM (**pamService**); this is the name of the configuration file to use in **/etc/pam.d**
- Whether to require a secure connection (**pamSecure**)
- Whether to fall back to LDAP authentication if PAM authentication fails (**pamFallback**)

```
pamFallback: false
pamSecure: false
pamService: ldapserver
```

14.10.2. Configuring PAM Pass-Through Authentication

NOTE

There can be multiple configuration instances of the PAM Pass-Through Authentication plug-in. An instance of the PAM Pass-Through Authentication plug-in can be applied to a subset of user entries by using the ***pamFilter*** attribute to set an LDAP filter to search for the specific entries to use with the plug-in.

PAM pass-through authentication is configured through the command line.

1. Make sure the PAM service is fully configured.
2. Remove the **pam_fprintd.so** module from the PAM configuration file.

IMPORTANT

The **pam_fprintd.so** module cannot be in the configuration file referenced by the ***pamService*** attribute of the PAM Pass-Through Authentication Plug-in configuration. Using the PAM **fprintd** module causes the Directory Server to hit the max file descriptor limit and can cause the Directory Server process to abort.

3. Enable the plug-in; this is disabled by default.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=PAM Pass-Through Auth Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

4. Create the PAM Pass-Through Auth plug-in configuration entry.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=Admin PAM PTA Config,cn=PAM Pass-Through Auth Plugin,cn=plugins,cn=config
cn: AD PAM PTA Config
```

5. Add the attributes available for the PAM plug-in. The available attributes are listed in Table 14.8, "PAM Pass-Through Auth Plug-in Attributes", and Example 14.1, "Example PAM Pass-Through Authentication Configuration Entry" has an example entry.

6. Restart the server to load the new plug-in configuration.

```
service dirsrv restart
```

Example 14.1. Example PAM Pass-Through Authentication Configuration Entry

```
dn: cn=Admin PAM PTA Config,cn=PAM Pass Through Auth,cn=plugins,cn=config
objectclass: top
objectclass: pamConfig
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: Admin PAM PTA Config
pamMissingSuffix: ALLOW
pamExcludeSuffix: cn=config
pamExcludeSuffix: o=NetscapeRoot
pamIDMapMethod: RDN ENTRY
pamIDAttr: customPamUid
pamFilter: (manager=uid=bjensen,ou=people,dc=example,dc=com)
pamFallback: FALSE
pamSecure: TRUE
pamService: ldapserver
```

14.10.3. Using PAM Pass-Through Authentication with Active Directory as the Backend

PAM pass-through authentication forwards the credentials from the Directory Server to the PAM service. One option is to set up and configure PAM modules specifically for Directory Server. Another option — and one which may be more repeatable and more convenient in some infrastructures — is to use the System Security Services Daemon (SSSD) to configure PAM. Because SSSD can use a variety of different identity stores, a lot of different servers or services can be used to provide credentials, including Active Directory.

Using pass-through authentication through SSSD is a daisy chain of services. The PAM PTA Plug-in is configured as normal. It points to the given PAM service file to use. This service file is managed by SSSD, and SSSD is configured to connect with whatever identity provider is required, even multiple providers.

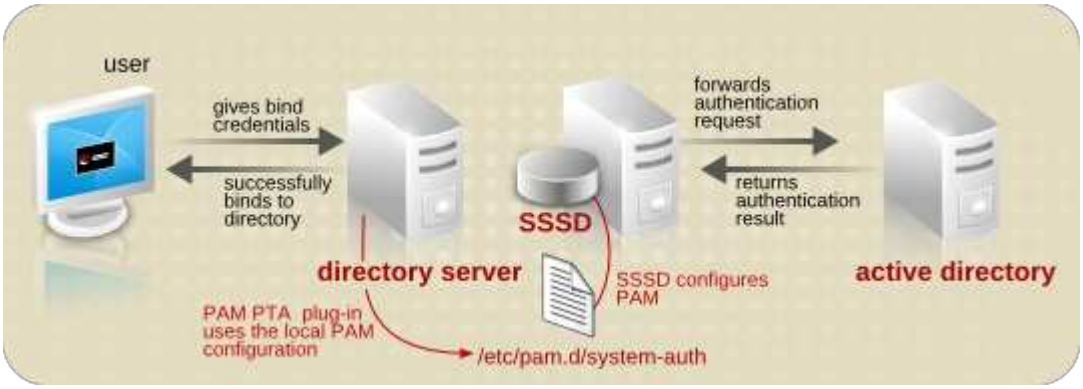


Figure 14.4. PAM Pass-Through Authentication with SSSD

For more information on SSSD, see the Red Hat Enterprise Linux 6 Deployment Guide .

To configure PAM pass-through authentication with Active Directory:

1. Configure SSSD to use the Active Directory server as one of its identity providers.

   This configuration is covered in the Red Hat Enterprise Linux 6 Deployment Guide .

2. Enable the PAM Pass-Through Auth plug-in; this is disabled by default.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=PAM Pass-Through Auth Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

3. Create the PAM Pass-Through Auth plug-in configuration entry.

```
ldapmodify -a -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=AD PAM PTA Config,cn=PAM Pass-Through Auth Plugin,cn=plugins,cn=config
cn: AD PAM PTA Config
```

4. Set the ***pamService*** attribute to point to the PAM configuration file managed by SSSD. By default, this is **/etc/pam.d/system-auth**.

```
pamService: system-auth
```

IMPORTANT

The **pam_fprintd.so** module cannot be in the configuration file referenced by the ***pamService*** attribute of the PAM Pass-Through Authentication Plug-in configuration. Using the PAM **fprintd** module causes the Directory Server to hit the max file descriptor limit and can cause the Directory Server process to abort.

5. Configure the ID map method and attribute. There are several options for how this can be done, depending on the Directory Server environment.

   The simplest is to use the RDN map method, which automatically uses the ***uid*** attribute (or the correct naming attribute) to map Directory Server users back to Active Directory users (since Active Directory is the identity provider).

```
pamIDMapMethod: RDN
```

   Similarly, this can be accomplished with the ENTRY map method by using the ***samAccountName*** attribute. If the user accounts in Directory Server are created with ***uid***s that match the ***samAccountName*** value for the user account in Active Directory, then the mapping is successful.

```
pamIDMapMethod: ENTRY
pamIDAttr: samAccountName
```

   If Windows synchronization is configured, then the ENTRY method can be used with the ***ntUserDomainId*** attribute. The Directory Server and Active Directory user accounts are already synced, based on that attribute value, so the PAM mapping is successful.

```
pamIDMapMethod: ENTRY
pamIDAttr: ntUserDomainId
```

6. Restart the server to load the plug-in configuration.

```
service dirsrv restart
```

14.11. MANUALLY INACTIVATING USERS AND ROLES

A single user account or set of accounts can be temporarily inactivated. Once an account is inactivated, a user cannot bind to the directory. The authentication operation will fail.

Users and roles are inactivated using the operational attribute ***nsAccountLock***. When an entry contains the ***nsAccountLock*** attribute with a value of **true**, the server rejects the bind.

The same procedures are used to inactivate users and roles. However, when a role is inactivated, the *members of the role* are inactivated, not the role entry itself. For more information about roles in general and how roles interact with access control in particular, see Chapter 6, *Organizing and Grouping Entries*.

WARNING

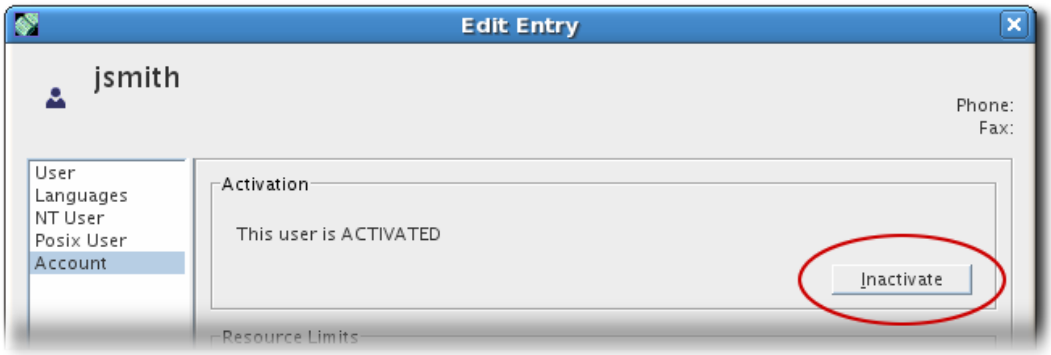
The root entry (the entry corresponding to the root or sub suffix) on a database cannot be inactivated. Chapter 3, *Creating Directory Entries* has information on creating the entry for a root or sub suffix, and Chapter 2, *Configuring Directory Databases* has information on creating root and sub suffixes.

14.11.1. Activating and Inactivating Users and Roles Using the Console

All user and role entries are active by default. They must be manually marked inactive and, once inactivated, must be manually re-activated.

1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the entry to inactivate.

   The **Edit Entry** dialog box appears.
3. Click **Account** in the left pane. The right pane states that the role or user is activate. Click the **Inactivate** button to inactivate the user or role (or the **Activate** button, to re-enable the entry).

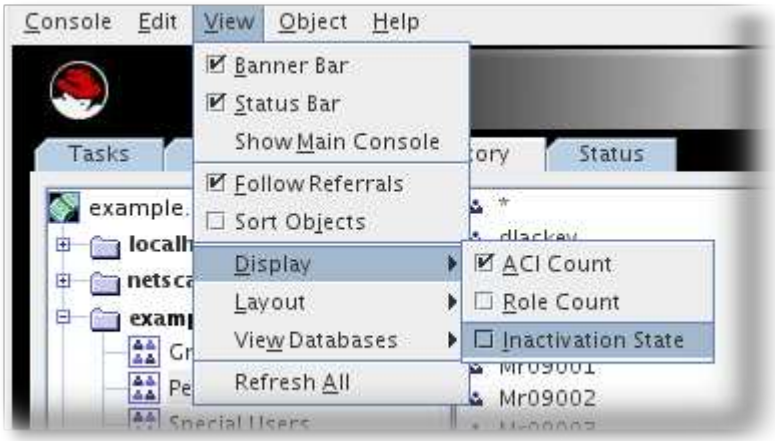


4. Click **OK**.

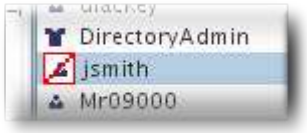
Alternatively, highlight the entry and select **Inactivate** (or **Activate**, if appropriate) from the **Object** menu.

14.11.2. Viewing Inactive Users and Roles

1. Select the **View** menu, and select the **Display** item.
2. Select the **Inactivation State** item.



When the inactivation state is visible, any inactive object is listed in the right pane of the Console with a red slash through it.



14.11.3. Inactivating and Activating Users and Roles Using the Command Line

The Directory Server uses dual scripts to inactivate or activate entries through the command line. The **ns-inactivate.pl** and **ns-activate.pl** script share similar options to identify the entry to modify, as listed in Table 14.10, "ns-inactivate.pl and ns-activate.pl Options".

For example, to inactivate a user account:

```
[root@server ~]# /usr/lib[64]/dirsrv/slapd-example/ns-inactivate.pl -D Directory Manager -w secret -p 389 -h
example.com  -I "uid=jfrasier,ou=people,dc=example,dc=com"
```

Then, the account can be re-activated:

```
[root@server ~]# /usr/lib[64]/dirsrv/slapd-example/ns-activate.pl -D Directory Manager -w secret -p 389 -h
example.com  -I "uid=jfrasier,ou=people,dc=example,dc=com"
```

Table 14.10. ns-inactivate.pl and ns-activate.pl Options

Option Name	Description
-D	The DN of the directory administrator.
-w	The password of the directory administrator.
-p	Port used by the server.
-h	Name of the server on which the directory resides.
-I	DN of the user account or role to inactivate or activate, depending on the script.

For more information about running the **ns-inactivate.pl** and **ns-activate.pl** scripts, see the *Directory Server Configuration and Command-Line Tool Reference*.

CHAPTER 15. MONITORING SERVER AND DATABASE ACTIVITY

This chapter describes monitoring database and Red Hat Directory Server logs. For information on using SNMP to monitor the Directory Server, see Chapter 16, *Monitoring Directory Server Using SNMP* .

15.1. TYPES OF DIRECTORY SERVER LOG FILES

Directory Server provides three types of logs to help better manage the directory and tune performance:

- The access contains information on client connections and connection attempts to the Directory Server instance.
- The error log contains detailed messages of errors and events the directory experiences during normal operations.

WARNING

If the Directory Server fails to write to the errors log, the server sends the message to **syslog** and exits.

- The audit log records changes made to each database as well as to server configuration. This log is not enabled by default.

15.2. VIEWING LOG FILES

The access and error logs are enabled by default and can be viewed immediately. Before the audit log can be viewed, audit logging must be enabled for the directory, or the audit log will not be kept.

NOTE

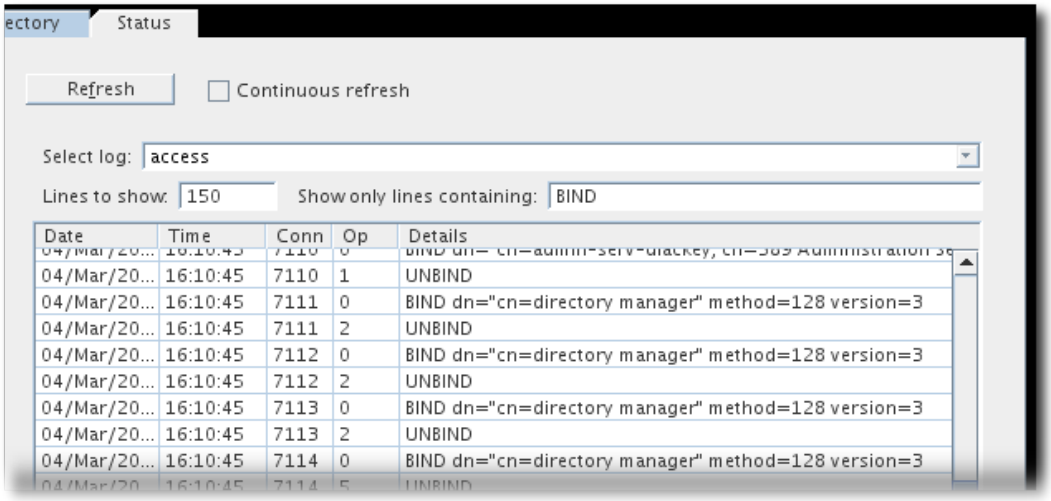
When the server is not running, the log files cannot be viewed in the Directory Server Console, but they can be viewed in Admin Express. Open the Admin Server URL in a browser:

```
http://hostname:admin_server_port
```

Then log in with the admin login ID and password, and click the link for **Administration Express**.

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Logs** folder. There are three folders available, for the access, error, and audit logs.

3. When you select the log type to view, a table displays a list of the last 25 entries in the selected log.
4. Optionally, change the settings of the log display and click **Refresh** to update the display.



- The **Select Log** pull-down menu allows you to select an archived (rotated) log rather than the currently active log.
- The **Lines to show** text box changes the number of log entries to display in the window.
- The **Show only lines containing** text box sets a filter, including regular expressions, to use to display only certain matching log entries.

NOTE

Selecting the **Continuous** check box refreshes the log display automatically every ten seconds.= Continuous log refresh does not work well with log files over 10 megabytes.

15.3. CONFIGURING LOG FILES

For all types of log files, the log *creation* and log *deletion* policies have to be configured. The log creation policy sets when a new log file is started, and the log deletion policy sets when an old log file is deleted.

15.3.1. Enabling or Disabling Logs

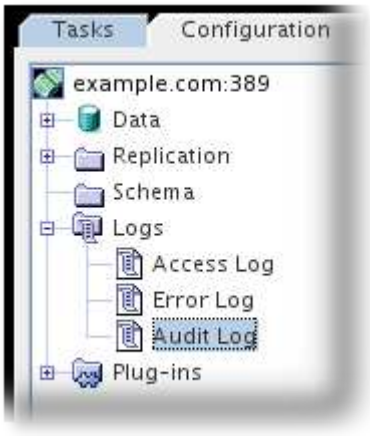
The access and error logging is enabled by default. However, audit logging is disabled by default.

NOTE

Disabling the access logging can be useful in some scenarios, because every 2000 accesses to the directory increases the log file by approximately 1 megabyte. However, before turning off access logging, consider that this information can help troubleshooting problems.

Enabling or Disabling Logging in the Directory Server Console

1. Log in to the Directory Server Console.
2. Select the **Configuration** tab.
3. In the navigation tree, expand the **Logs** folder, and select the folder for the log to enable or disable.



4. To enable or disable logging, select the **Enable Logging** check box.

5. If the log is being enabled, enter the full path and file name for the Directory Server to use for logging in the field provided. The default path is **/var/log/dirsrv/slapd-*instance*/***log_type***, such as **/var/log/dirsrv/slapd-*instance*/access**.

6. Click **Save**.

Enabling or Disabling Logging Using the Command Line

You can use the **ldapmodify** utility to modify the parameters in the **cn=config** subtree that control the Directory Server logging feature:

- Access log: ***nsslapd-accesslog-logging-enabled***
- Error log: ***nsslapd-errorlog-logging-enabled***
- Audit log: ***nsslapd-auditlog-logging-enabled***

For further details, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to enable audit logging, enter:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-auditlog-logging-enabled
nsslapd-auditlog-logging-enabled: on
```

15.3.2. Defining a Log File Rotation Policy

To periodically archive the current log file and create a new one, set a log file rotation policy. You can update the settings in the **cn=config** subtree using the Directory Server Console or command line.

You can set the following configuration parameters to control the log file rotation policy:

Access mode

The access mode sets the file permissions on newly created log files.

- Access log: ***nsslapd-accesslog-mode***
- Error log: ***nsslapd-errorlog-mode***
- Audit log: ***nsslapd-auditlog-mode***

Maximum number of logs

Sets the maximum number of log files to keep. When the number of files is reached, Directory Server deletes the oldest log file before creating the new one.

- Access log: ***nsslapd-accesslog-maxlogsperdir***
- Error log: ***nsslapd-errorlog-maxlogsperdir***

- Audit log: ***nsslapd-auditlog-maxlogsperdir***

File size for each log

Sets the maximum size of a log file in megabytes before it is rotated.

- Access log: ***nsslapd-accesslog-maxlogsize***
- Error log: ***nsslapd-errorlog-maxlogsize***
- Audit log: ***nsslapd-auditlog-maxlogsize***

Create a log every

Sets the maximum age of a log file.

- ***nsslapd-accesslog-logrotationtime*** and ***nsslapd-auditlog-logrotationtimeunit***
- ***nsslapd-errorlog-logrotationtime*** and ***nsslapd-errorlog-logrotationtimeunit***
- ***nsslapd-auditlog-logrotationtime*** and ***nsslapd-auditlog-logrotationtimeunit***

Additionally, you can set the time when the log file is rotated using the following parameters:

- ***nsslapd-accesslog-logrotationsynchour*** and ***nsslapd-accesslog-logrotationsyncmin***
- ***nsslapd-errorlog-logrotationsynchour*** and ***nsslapd-errorlog-logrotationsyncmin***
- ***nsslapd-auditlog-logrotationsynchour*** and ***nsslapd-auditlog-logrotationsyncmin***

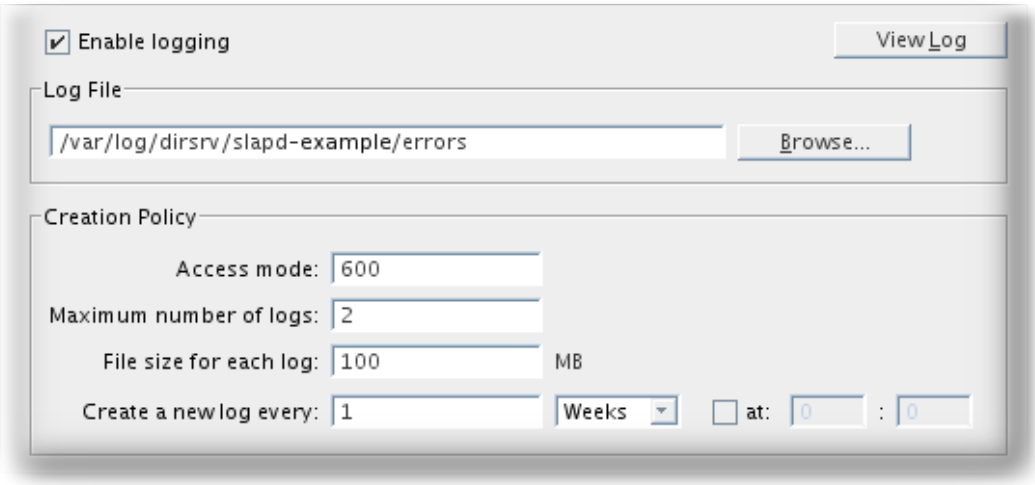
For details, see the parameter descriptions in the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

Each log file starts with a title, which identifies the server version, host name, and port, for ease of archiving or exchanging log files. For example:

```
389-Directory/1.2.11.15 B2016.312.1929
server.example.com:389 (/etc/dirsrv/slapd-instance)
```

Configuring Log File Rotation in the Directory Server Console

1. Log in to the Directory Server Console.
2. Select the **Configuration** tab.
3. In the navigation tree, expand the **Logs** folder, and select the folder for the log you want to update the settings.
4. Set the logging settings in the **Creation policy** area. For example:



5. Click **Save**.

Configuring Log File Rotation Using the Command Line

You can use the **ldapmodify** utility to modify the parameters controlling the Directory Server logging features. For example for the error log, to set access mode **600**, to keep maximum **2**, and to rotate log files at a size of **100** MB or every **5 days**, run:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-errorlog-mode
nsslapd-errorlog-mode: 600
-
replace: nsslapd-errorlog-maxlogsperdir
nsslapd-errorlog-maxlogsperdir: 2
-
replace: nsslapd-errorlog-maxlogsize
nsslapd-errorlog-maxlogsize: 100
-
replace: nsslapd-errorlog-logrotationtime
nsslapd-errorlog-logrotationtime: 5
-
replace: nsslapd-errorlog-logrotationtimeunit
nsslapd-errorlog-logrotationtimeunit: day
```

15.3.3. Defining a Log File Deletion Policy

Directory Server automatically deletes old archived log files, if you set a **Deletion Policy**.

NOTE

You can only set a log file deletion policy if you have a log file rotation policy set. Directory Server applies the deletion policy at the time of log rotation.

You can set the following configuration parameters to control the log file deletion policy:

Total log size

If the size of all access, error, or audit log files increases the configured value, the oldest log file is automatically deleted.

- Access log: ***nsslapd-accesslog-logmaxdiskspace***
- Error log: ***nsslapd-errorlog-logmaxdiskspace***
- Audit log: ***nsslapd-auditlog-logmaxdiskspace***

Free disk space is less than

When the free disk space reaches this value, the oldest archived log file is automatically deleted.

- Access log: ***nsslapd-accesslog-logminfreediskspace***
- Error log: ***nsslapd-errorlog-logminfreediskspace***
- Audit log: ***nsslapd-auditlog-logminfreediskspace***

When a file is older than a specified time

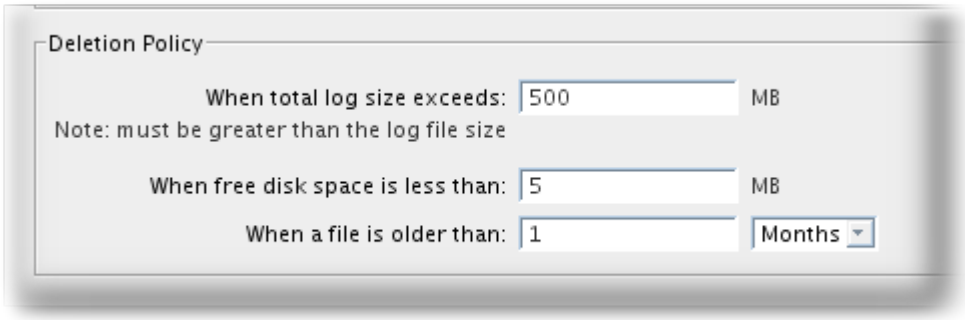
When a log file is older than the configured time, it is automatically deleted.

- Access log: ***nsslapd-accesslog-logexpirationtime*** and ***nsslapd-accesslog-logexpirationtimeunit***
- Error log: ***nsslapd-errorlog-logminfreediskspace*** and ***nsslapd-errorlog-logexpirationtimeunit***
- Audit log: ***nsslapd-auditlog-logminfreediskspace*** and ***nsslapd-auditlog-logexpirationtimeunit***

For further details, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

Configuring a Log Deletion Policy in the Directory Server Console

1. Log in to the Directory Server Console.
2. Select the **Configuration** tab.
3. In the navigation tree, expand the **Logs** folder, and select the folder for the log you want to update the settings.
4. Set the logging settings in the **Deletion Policy** area. For example:



5. Click **Save**.

Configuring Log Deletion Policy Using the Command Line

You can use the **ldapmodify** utility modify the parameters controlling the Directory Server logging features. For example, to auto-delete the oldest access log file if the total size of all access log files increases **500** MB, run:

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-accesslog-logmaxdiskspace
nsslapd-accesslog-logmaxdiskspace: 500
```

15.3.4. Manual Log File Rotation

The Directory Server supports automatic log file rotation for all three logs. However, it is possible to rotate log files manually if there are no automatic log file creation or deletion policies configured. By default, access, error, and audit log files can be found in the following location:

```
/var/log/dirsrv/slapd-instance
```

To rotate log files manually:

1. Shut down the server.

```
systemctl stop dirsrv.target instance
```

2. Move or rename the log file being rotated so that the old log file is available for future reference.
3. Restart the server.

```
systemctl restart dirsrv.target instance
```

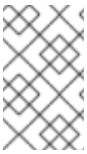
15.3.5. Configuring Log Levels

Both the access and the error log can record different amounts of information, depending on the log level that is set.

You can set the following configuration parameters to control the log levels for the:

- Access log: ***nsslapd-accesslog-level***
- Error log: ***nsslapd-errorlog-level***

For further details and a list of the supported log levels, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

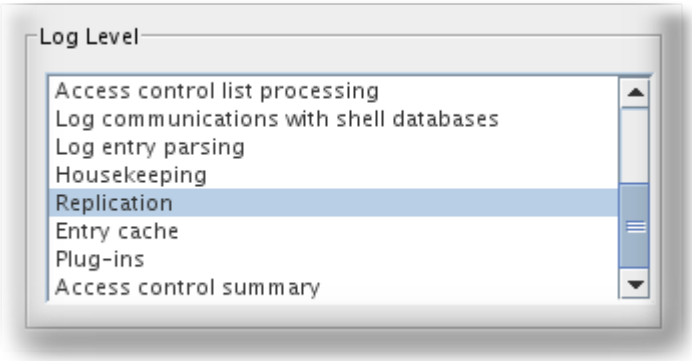


NOTE

Changing the log level from the default can cause the log file to grow very rapidly. Red Hat recommends not to change the default values without being asked to do so by the Red Hat technical support.

Configuring the Log Level in the Directory Server Console

1. Log in to the Directory Server Console.
2. Select the **Configuration** tab.
3. In the navigation tree, expand the **Logs** folder, and select the folder for the log you want to update the settings.
4. Set the log level in the **Log Level** area. For example, for the error log file



5. Click **Save**.

Configuring the Log Level Using the Command Line

You can use the **ldapmodify** utility to set the log level. For example, to enable search filter logging (**32**) and config file processing (**64**), set the ***nsslapd-errorlog-level*** parameter to **96** (32 + 64):

```
[root@server ~]# ldapmodify -D "cn=directory manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 96
```

15.4. GETTING ACCESS LOG STATISTICS

The **logconv.pl** script parses the access log and returns summary information on different users and operations that have been run on the server.

At its simplest, the script simply parses the access log (or logs):

```
# logconv.pl /relative/path/to/accessLog
```

The script can accept wildcards to parse multiple access logs, which is useful if log rotation is used.

```
# logconv.pl /var/log/dirsrv/slapd-instance/access*
```

The different options for **logconv.pl** are covered in the manpage and in the *Configuration and Command-Line Tool Reference*.

There are several different ways that **logconv.pl** can be used to pull general usage information from the access logs.

At its simplest, **logconv.pl** prints a list of total operations, total number of connections, counts per each operation type, counts for some extended operations like persistent searches, and bind information.

```
# logconv.pl /var/log/dirsrv/slapd-instance/access
Access Log Analyzer 8.1
Command: logconv.pl /var/log/dirsrv/slapd-instance/access
Processing 1 Access Log(s)...
```

```
[001] /var/log/dirsrv/slapd-instance/access size (bytes):       141640

Total Log Lines Analysed:  1056

---------- Access Log Output ------------

Start of Logs:    18/Nov/2016:10:43:55
End of Logs:      18/Nov/2016:12:25:02

Processed Log Time:  1 Hours, 41 Minutes, 7 Seconds

Restarts:                    3
Total Connections:           29
 - LDAP Connections:         29
 - LDAPI Connections:        0
 - LDAPS Connections:        0
 - StartTLS Extended Ops:    0
Peak Concurrent Connections:  5
Total Operations:            481
Total Results:               479
Overall Performance:         99.6%

Searches:                    326          (0.05/sec)  (3.22/min)
Modifications:               15           (0.00/sec)  (0.15/min)
Adds:                        107          (0.02/sec)  (1.06/min)
Deletes:                     0            (0.00/sec)  (0.00/min)
Mod RDNs:                    0            (0.00/sec)  (0.00/min)
Compares:                    0            (0.00/sec)  (0.00/min)
Binds:                       31           (0.01/sec)  (0.31/min)

Proxied Auth Operations:     0
Persistent Searches:         0
Internal Operations:         0
Entry Operations:            0
Extended Operations:         0
Abandoned Requests:          0
Smart Referrals Received:    0

VLV Operations:              2
VLV Unindexed Searches:      0
VLV Unindexed Components:    2
SORT Operations:             14

Entire Search Base Queries:  42
Paged Searches:              0
Unindexed Searches:          0
Unindexed Components:        41

FDs Taken:                   29
FDs Returned:                29
Highest FD Taken:            69

Broken Pipes:                0
Connections Reset By Peer:   0
Resource Unavailable:        0
Max BER Size Exceeded:       0

Binds:                       31
Unbinds:                     24
 - LDAP v2 Binds:            0
 - LDAP v3 Binds:            31
 - AUTOBINDs:                0
 - SSL Client Binds:         0
 - Failed SSL Client Binds:  0
 - SASL Binds:               0
 - Directory Manager Binds:  20
 - Anonymous Binds:          8
 - Other Binds:              3

Cleaning up temp files...
Done.
```

In addition to the summary information for operations and connections, more detailed summary information for all of the connections to the server. This information includes things like most common IP addresses used to connect to the server, DNs with the most failed login attempts, total bind DNs used to access the server, and the most common error or return codes.

Additional connection summaries are passed as a single option. For example, listing the number of DNs used to connect to the server (**b**) and the total connection codes returned by the server (**c**) are passed as **-bc**.

```
# logconv.pl -bc /var/log/dirsrv/slapd-instance/access
...
----- Total Connection Codes -----

U1              3    Cleanly Closed Connections
B1              1    Bad Ber Tag Encountered

----- Top 20 Bind DN's -----

Number of Unique Bind DN's: 212

1801            cn=directory manager
1297            Anonymous Binds
311             uid=jsmith,ou=people...
87              uid=bjensen,ou=peopl...
85              uid=mreynolds,ou=peo...
69              uid=jrockford,ou=peo...
55              uid=sspencer,ou=peop...
...
```

The data can be limited to entries after a certain start time (**-S**), before a certain end time (**-E**), or within a range. When start and end times are set, the **logconv.pl** first prints the time range given, then the summary for that period.

```
# logconv.pl -S "[01/Jul/2016:16:11:47.000000000 -0400]" -E "[01/Jul/2016:17:23:08.999999999 -0400]"
/var/log/dirsrv/slapd-instance/access
...
----------- Access Log Output ------------

Start of Logs:    01/Jul/2016:16:11:47
End of Logs:      01/Jul/2016:17:23:08
...
```

The start and end period only sets time limits for the data used to generate the total summary counts. It still shows aggregated, or total, counts. To get a view of the patterns in connections and operations to the Directory Server, it is possible to output data with counts per minute (**-M**) or per second (**-m**). In this case, the data are printed, in time unit increments, to a specified CSV output file.

```
# logconv.pl -m|-M outputFile accessLogFile
```

For example:

```
# logconv.pl -M /home/output/statsPerMin.txt /var/log/dirsrv/slapd-instance/access*
```

The **-M|-m** options can also be used with the **-S** and **-E** arguments, to get per-minute or per-second counts within a specific time period.

Each row in the file represents one unit of time, either minute or second, with total counts for that time period. The CSV file (for both per-minute and per-second statistics) contains the following columns, in order:

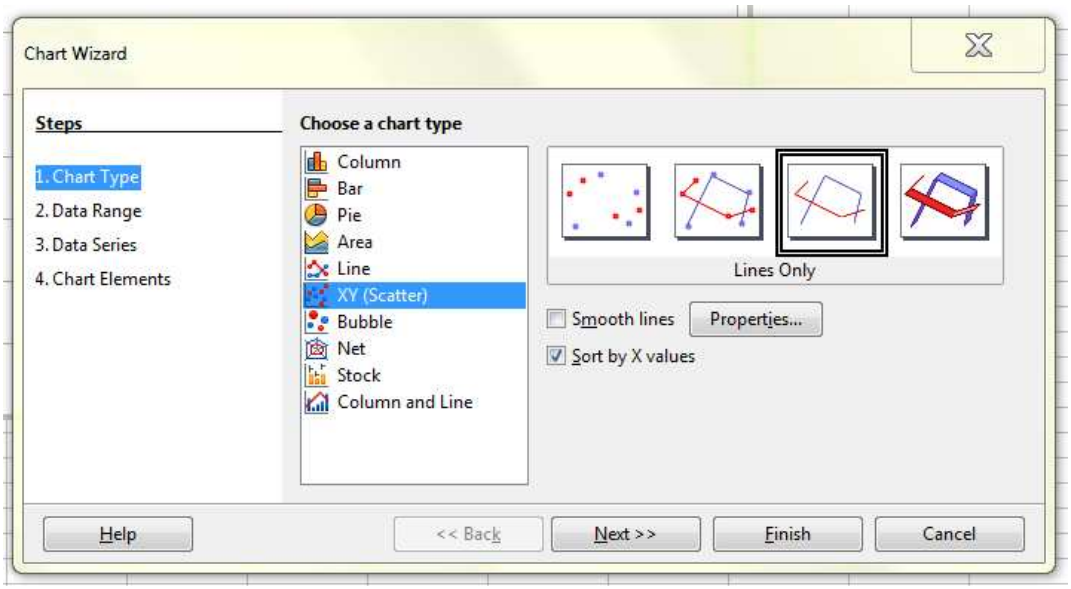
```
Time,time_t,Results,Search,Add,Mod,Modrdn,Delete,Abandon,Connections,SSL Conns,Bind,Anon
Bind,Unbind,Unindexed
```

The CSV file can be manipulated in any spreadsheet program, like OpenOffice Calc, and in many other business applications. The procedures for importing the CSV data and generating charts or other metrics depends on the application itself.

For example, to create a chart in OpenOffice Calc:

1. Open the CSV file.
2. Click the **Insert** menu, and select **Chart**.

3. In the **Chart Type** area, set the chart type to **XY (Scatter)**.
    1. Set the subtype to lines only.
    2. Select the option to sort by X values.



4. Accept the defaults in the other screens (particularly, to use the data series in columns and to set the first row and first column as labels), and create the chart.

15.5. REPLACING LOG FILES WITH A NAMED PIPE

The named pipe log script enables administrators to replace a log file with a named pipe to automatically process the log data. This provides advanced logging features, such as:

- Log only certain events, such as failed binds or connections from certain IP addresses.
- Log only lines that match a regular expression.
- Log only a defined number of lines.
- Send an email or other notification when a defined event is logged.

You can configure a named pipe for logging:

- For testing purposes, see Section 15.5.1, "Temporarily Replacing a Log File with a Named Pipe"
- For permanent usage, see Section 15.5.2, "Creating a New Named Pipe for Logging"

15.5.1. Temporarily Replacing a Log File with a Named Pipe

If you replace a log file with a named pipe, no server modifications are required. With this configuration, you cannot use log viewers, such as in the Admin Console, because they require to read the content a from a file.

To replace a log file with a named pipe:

1. Stop the Directory Server instance:

```
# systemctl stop dirsrv.target
```

2. Remove the log file. For example:

```
# rm -f /var/log/dirsrv/errors
```

3. Configure the named pipe to start with the Directory Server. For details, see Section 15.5.3, "Starting and Shutting Down the Named Pipe with the Directory Server Service".
4. Start the Directory Server instance:

```
# systemctl start dirsrv.target
```

IMPORTANT

When the log files are rotated, the named pipe is replaced with a regular file. Use this procedure only as a temporary solution. For a permanent solution, see Section 15.5.2, "Creating a New Named Pipe for Logging"

15.5.2. Creating a New Named Pipe for Logging

To log to a named pipe and additionally be able to use log viewers, such as in the Admin Console, configure the named pipe to use a different name than the name of your log file:

1. Configure the named pipe to start with Directory Server. To enable log viewers, additionally redirect the output of the **ds-logpipe.py** command to a file. For example:

   ```
   python /usr/bin/ds-logpipe.py ... > /var/log/dirsrv/slapd-instance/errors &
   ```

   For details, see Section 15.5.3, "Starting and Shutting Down the Named Pipe with the Directory Server Service".

2. Update the Directory Server configuration to log to the named pipe. For example, to send the access log to the **/var/log/dirsrv/slapd-instance/access.pipe** named pipe:

   ```
   # ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
   dn: cn=config
   changetype: modify
   replace: nsslapd-accesslog
   nsslapd-accesslog: /var/log/dirsrv/slapd-instance/access.pipe
   ```

   Optionally, you can also set:

   - The ***nsslapd-errorlog*** parameter for error events.
   - The ***nsslapd-auditlog*** parameter for audit events. Note that audit logging is disabled by default. To enable it, additionally set the ***nsslapd-accesslog-logging-enabled*** parameter to **on**.

   NOTE

   Updating the parameters takes effect immediately. However, you must start the named pipe manually or restart the Directory Server instance.

3. Disable buffering and log rotation for the event you configured the named pipe for. For example, to disable the features for the access log:

   ```
   # ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x
   dn: cn=config
   changetype: modify
   replace: nsslapd-accesslog-logbuffering
   nsslapd-accesslog-logbuffering: off
   -
   replace: nsslapd-accesslog-maxlogsperdir
   nsslapd-accesslog-maxlogsperdir: 1
   -
   replace: nsslapd-accesslog-logexpirationtime
   nsslapd-accesslog-logexpirationtime: -1
   -
   replace: nsslapd-accesslog-logrotationtime
   nsslapd-accesslog-logrotationtime: -1
   ```

   To disable the parameters for the error log, update:

   - ***nsslapd-errorlog-logbuffering***
   - ***nsslapd-errorlog-maxlogsperdir***
   - ***nsslapd-errorlog-logexpirationtime***
   - ***nsslapd-errorlog-logrotationtime***

To disable the parameters for the audit log, update:

- ***nsslapd-auditlog-logbuffering***
- ***nsslapd-auditlog-maxlogsperdir***
- ***nsslapd-auditlog-logexpirationtime***
- ***nsslapd-auditlog-logrotationtime***

4. Restart the Directory Server instance to start the pipe.

```
# systemctl restart dirsrv.target
```

15.5.3. Starting and Shutting Down the Named Pipe with the Directory Server Service

To start and shut down the named pipe with the Directory Server instance:

1. Open the **/etc/sysconfig/dirsrv-*instance*** instance configuration file.

WARNING

Do not edit the **/etc/sysconfig/dirsrv** file.

2. Append the **ds-logpipe.py** commands at the end of the file. For example:

```
# Only keep the last 1000 lines of the error log and
# additionally redirect all log data to the
# /var/log/dirsrv/slapd-instance/errors file
python /usr/bin/ds-logpipe.py /var/log/dirsrv/slapd-instance/errors.pipe -m 1000 -u dirsrv -s
/var/run/dirsrv/slapd-instance.pid > /var/log/dirsrv/slapd-instance/errors &

# Only log failed binds
python /usr/bin/ds-logpipe.py /var/log/dirsrv/slapd-instance/access.pipe -u dirsrv -s
/var/run/dirsrv/slapd-instanec.pid --plugin=/usr/share/dirsrv/data/failedbinds.py
failedbinds.logfile=/var/log/dirsrv/slapd-instance/access.failedbinds &
```

For details, see the ds-logpipe.py(1) man page.

IMPORTANT

Make sure that each named pipe command ends with an ***&*** sign to send the **ds-logpipe.py** process to the background.

15.5.4. Using Plug-ins with the Named Pipe Log

You can call a plug-in to read the log data from the named pipe to perform operations on the log data. When using plug-ins with the named pipe log script, consider the following:

- The plug-in function is called for every line read from the named pipe.
- The plug-in function must be a Python script and use the **.py** suffix.
- Any plug-in arguments are passed in the command line to the **ds-logpipe.py** named pipe log script.
- A **pre** operation function can be called for when the plug-in is loaded.
- A **post** operation function can be called for when the plug-in exits.

15.5.4.1. Loading Plug-ins with the Named Pipe Log Script

There are two options for the **ds-logpipe.py** command to use with plug-ins:

- The **--plugin** option gives the path to the plug-in file.

- The *plugin.arg* option passes plug-in arguments to the named pipe log script.
  - **plugin**: The file name without the **.py** suffix.
  - **arg**: Any argument allowed in the plug-in.

For example:

```
ds-logpipe.py /var/log/dirsrv/slapd-example/errors.pipe --plugin=/usr/share/dirsrv/data/example-funct.py example-funct.regex="warning" > /var/log/dirsrv/warnings.txt
```

If there are more than one value passed to the same argument, they are converted into a list of values in the plug-in. For example, this script sets two values for the **arg1** argument:

```
--plugin=/path/to/plugin_name.py plugin_name.arg1=example1 plugin_name.arg1=example2 plugin_name.arg2=demo
```

In the plug-in, this is converted to:

```
{'arg1': ['example1', 'example2'], 'arg2': 'demo'}
```

This is a Python **dictionary** object with two keys. The first key is the string **arg1**, and its value is a Python list object with two elements, the strings **foo** and **bar**. The second key is the string **arg2**, and its value is the string **baz**. If an argument has only a single value, it is left as a simple string. Multiple values for a single argument name are converted into a list of strings.

15.5.4.2. Writing Plug-ins to Use with the Named Pipe Log Script

The **ds-logpipe.py** command supports the following functions in a plug-in:

- **plugin()**: Mandatory. Code in this function is applied to every line of log data received.
- **pre()**: Optional. Code is run when the plug-in is started.
- **post()**: Optional. Code is run when the plug-in exits.

Each function can have any arguments defined for it, and these arguments can then be passed to the script using the *plugin.arg* option. Additionally, each function can have its own return values and actions defined for it.

Example 15.1. Simple Named Pipe Log Plug-in

```
def pre(myargs):
    retval = True
    myarg = myargs['argname']
    if isinstance(myarg, list): # handle list of values
    else: # handle single value
    if bad_problem:
        retval = False
    return retval

def plugin(line):
    retval = True
    # do something with line
    if something_is_bogus:
        retval = False
    return retval

def post(): # no arguments
    # do something
    # no return value
```

15.5.5. Troubleshooting the Named Pipe

15.5.5.1. Directory Server Hangs When Writing to the Named Pipe

If the **ds-logpipe.py** command terminates unexpectedly, the Directory Server hangs while writing to the named pipe. To fix the problem:

Restart the named pipe manually or if configured in the **/etc/sysconfig/dirsrv-*instance*** file, run:

```
# (. /etc/sysconfig/dirsrv-instance)
```

15.6. MONITORING THE LOCAL DISK FOR GRACEFUL SHUTDOWN

When the disk space available on a system becomes too small, the Directory Server process (**slapd**) crashes. Any abrupt shutdown runs the risk of corrupting the database or losing directory data.

It is possible to monitor the disk space available to the **slapd** process. A disk monitoring thread is enabled using the ***nsslapd-disk-monitoring*** configuration attribute. This creates a monitoring thread that wakes every ten (10) seconds to check for available disk space in certain areas.

If the disk space approaches a defined threshold, then the **slapd** begins a series of steps (by default) to reduce the amount of disk space it is consuming:

- Verbose logging is disabled.
- Access logging and error logging are disabled.
- Rotated (archived) logs are deleted.

NOTE

Error log messages are always recorded, even when other changes are made to the logging configuration.

If the available disk space continues to drop to half of the configured threshold, then the **slapd** begins a graceful shut down process (within a grace period); and if the available disk space ever drops to 4KB, then the **slapd** process shuts down immediately. If the disk space is freed up, then the shutdown process is aborted, and all of the previously disabled log settings are re-enabled.

By default, the monitoring thread checks the configuration, transaction log, and database directories. An additional attribute (***nsslapd-disk-monitoring-logging-critical***) can be set to include the logs directory when evaluating disk space.

Disk monitoring is disabled by default, but it can be enabled and configured by adding the appropriate configuration attributes to the **cn=config** entry. Table 15.1, “Disk Monitoring Configuration Attributes” lists all of the configuration options.

1. Using **ldapmodify**, add the disk monitoring attributes. At a minimum, turn on the ***nsslapd-disk-monitoring*** attribute to enable disk monitoring. The default threshold is 2MB; this can be configured (optionally) in the ***nsslapd-disk-monitoring-threshold*** attribute.

   For example:

   ```
   [jsmith@server ~]$ ldapmodify -D "cn=directory manager" -W -x
   dn: cn=config
   changetype: modify
   add: nsslapd-disk-monitoring
   nsslapd-disk-monitoring: on
   -
   add: nsslapd-disk-monitoring-threshold
   nsslapd-disk-monitoring-threshold: 3000000
   -
   add: nsslapd-disk-monitoring-grace-period
   nsslapd-disk-monitoring-grace-period: 20
   ```

2. Restart the Directory Server to load the new configuration.

   ```
   [root@server ~]# service dirsrv restart
   ```

Table 15.1. Disk Monitoring Configuration Attributes

Configuration Attribute	Description

Configuration Attribute	Description
nsslapd-disk-monitoring	Enabled disk monitoring. This is the only required attribute, since the other configuration options have usable defaults.
nsslapd-disk-monitoring-grace-period	Sets a grace period to wait before shutting down the server after it hits half of the disk space limit. This gives an administrator time to address the situation. The default value is 60 (minutes).
nsslapd-disk-monitoring-logging-critical	Sets whether to shut down the server if the log directories pass the halfway point set in the disk space limit. This prevents the monitoring thread from disabling audit or access logging or from deleting rotated logfiles.
nsslapd-disk-monitoring-threshold	Sets the amount of disk space, in bytes, to use to evaluate whether the server has enough available disk space. Once the space reaches half of this threshold, then the server begins a shut down process. The default value is 2000000 (2MB).

15.7. MONITORING SERVER ACTIVITY

The Directory Server's current activities can be monitored from either the Directory Server Console or the command line. It is also possible to monitor the activity of the caches for all of the database.

Many performance counters in Directory Server use 32-bit numbers. This includes many kind of performance monitoring, such as the number of active connections (***nsslapd-db-active-txns***), the number of operations (***threads***), and the number of search requests (***nsslapd-db-cache-try***). However, under heavy loads, these 32-bit counter numbers can wrap too quickly. The ***nsslapd-counters*** attribute enabled the Directory Server to use 64-bit values for counter numbers, even on 32-bit systems. This enables long term statistics gathering for high-traffic systems.

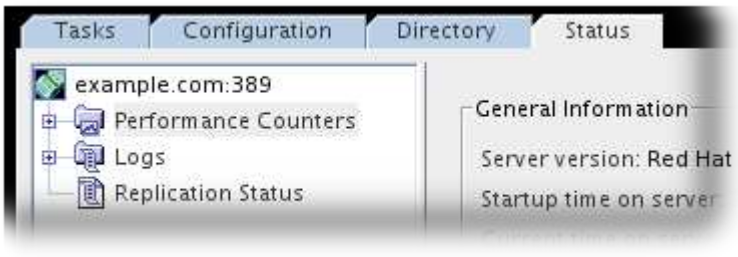
Some of the counters for Directory Server database attributes monitored by server use 64-bit integers, even on 32-bit systems:

- opsinitiated
- opscompleted
- entriessent
- bytessent
- totalconnections

The counters which use 64-bit integers are not configurable.

15.7.1. Monitoring the Server from the Directory Server Console

1. Select the **Status** tab.
2. In the navigation tree, select **Performance Counters**.



The **Status** tab in the right pane displays current information about server activity. If the server is currently not running, this tab will not provide performance monitoring information.

3. Click **Refresh** to refresh the current display. For the server to continuously update the displayed information, select the **Continuous** check box.

The **General Information** table shows basic information about the server, which helps set a baseline about the statistics that have been gathered.

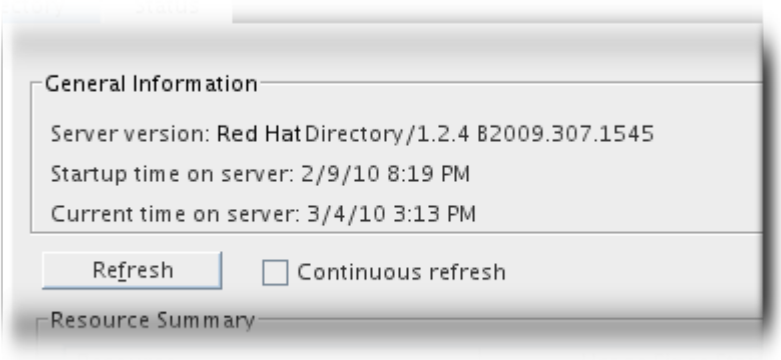


Table 15.2. General Information (Server)

Field	Description
Server Version	Identifies the current server version.
Startup Time on Server	The date and time the server was started.
Current Time on Server	The current date and time on the server.

The **Resource Summary** table shows the totals of all operations performed by that instance.

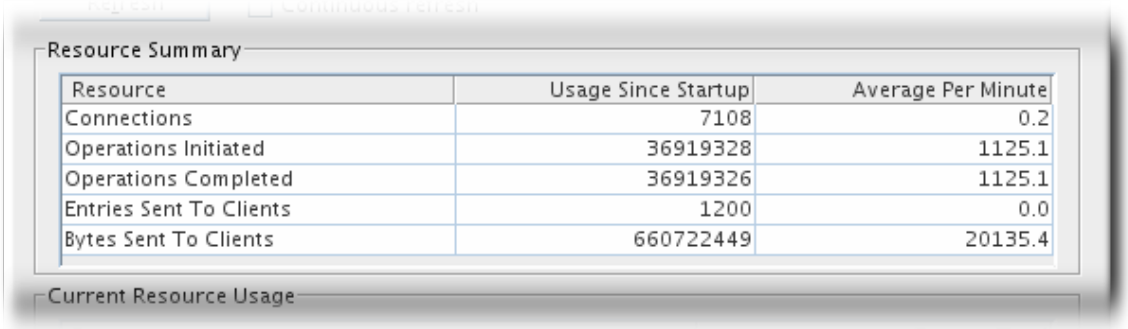


Table 15.3. Resource Summary

Resource	Usage Since Startup	Average Per Minute
Connections	The total number of connections to this server since server startup.	Average number of connections per minute since server startup.
Operations Initiated	The total number of operations initiated since server startup. Operations include any client requests for server action,such as searches, adds, and modifies. Often, multiple operations are initiated for each connection.	Average number of operations per minute since server startup.
Operations Completed	The total number of operations completed by the server since server startup.	Average number of operations per minute since server startup.

Resource	Usage Since Startup	Average Per Minute
Entries Sent to Clients	The total number of entries sent to clients since server startup. Entries are sent to clients as the result of search requests.	Average number of entries sent to clients per minute since server startup.
Bytes Sent to Clients	The total number of bytes sent to clients since server startup.	Average number of bytes sent to clients per minute since server startup.

The **Current Resource Usage** table shows the current demands on the server.

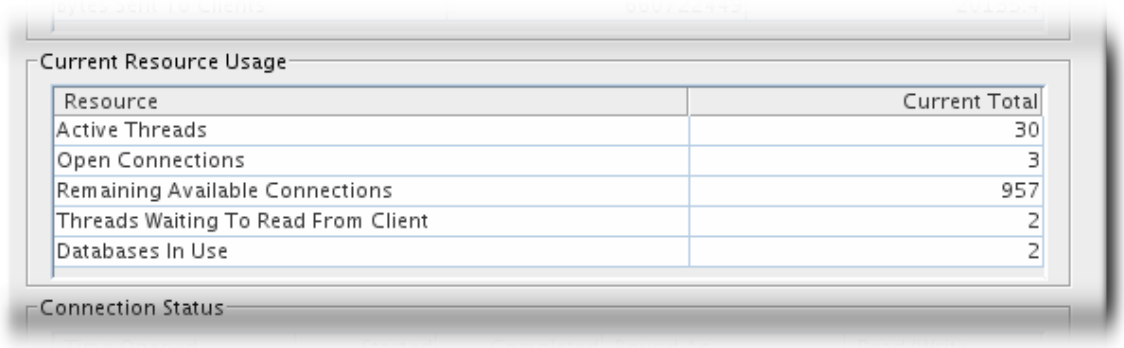
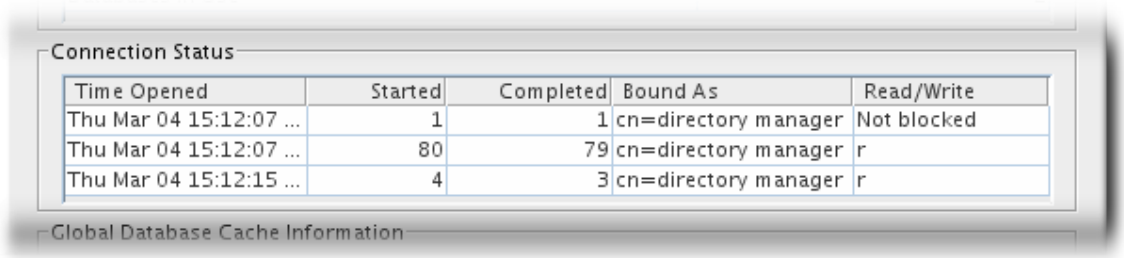


Table 15.4. Current Resource Usage

Resource	Current Total
Active Threads	The current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.
Open Connections	The total number of open connections. Each connection can account for multiple operations, and therefore multiple threads.
Remaining Available Connections	The total number of remaining connections that the server can concurrently open. This number is based on the number of currently open connections and the total number of concurrent connections that the server is allowed to open. In most cases, the latter value is determined by the operating system and is expressed as the number of file descriptors available to a task.
Threads Waiting to Write to Client	The total number of threads waiting to write to the client. Threads may not be immediately written when the server must pause while sending data to a client. Reasons for a pause include a slow network, a slow client, or an extremely large amount of information being sent to the client.
Threads Waiting to Read from Client	The total number of threads waiting to read from the client. Threads may not be immediately read if the server starts to receive a request from the client, and then the transmission of that request is halted for some reason. Generally, threads waiting to read are an indication of a slow network or client.
Databases in Use	The total number of databases being serviced by the server.

The **Connection Status** table simply lists the current active connections, with related connection information.


Connection Status

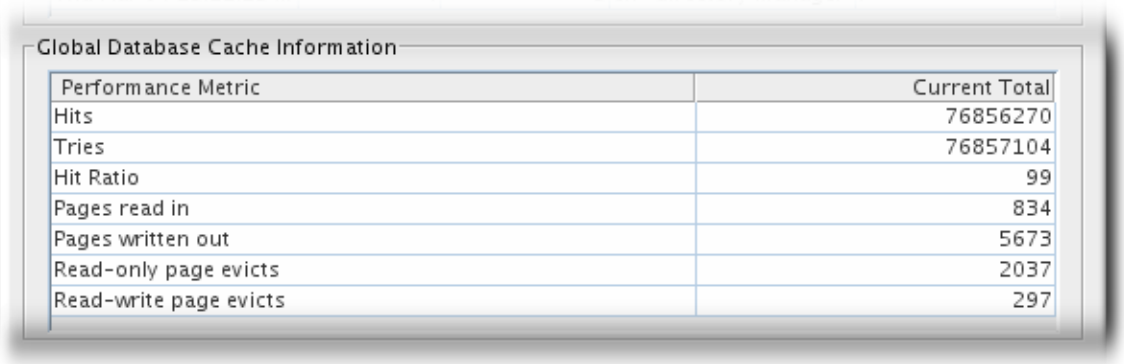
Time Opened	Started	Completed	Bound As	Read/Write
Thu Mar 04 15:12:07 ...	1	1	cn=directory manager	Not blocked
Thu Mar 04 15:12:07 ...	80	79	cn=directory manager	r
Thu Mar 04 15:12:15 ...	4	3	cn=directory manager	r

Global Database Cache Information

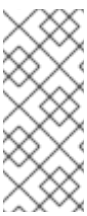
Table 15.5. Connection Status

Table Header	Description
Time Opened	The time on the server when the connection was initially opened.
Started	The number of operations initiated by this connection.
Completed	The number of operations completed by the server for this connection.
Bound as	The distinguished name used by the client to bind to the server. If the client has not authenticated to the server, the server displays **not bound** in this field.
Read/Write	Indicates whether the server is currently blocked for read or write access to the client. There are two possible values: Not blocked means that the server is idle,actively sending data to the client, or actively reading data from the client. Blocked means that the server is trying to send data to the client or read data from the client but cannot. The probable cause is a slow network or client.

The **Global Database Cache** table lists the cache information for all databases within the Directory Server instance.


Global Database Cache Information

Performance Metric	Current Total
Hits	76856270
Tries	76857104
Hit Ratio	99
Pages read in	834
Pages written out	5673
Read-only page evicts	2037
Read-write page evicts	297



NOTE

Although the performance counter for the global database cache is listed with the other server performance counters in the Directory Server Console, the actual database cache entries are located and monitored in **cn=monitor,cn=*database_instance*,cn=ldbm database,cn=plugins,cn=config**, as are the other database activities. Monitoring these entries through the command line is covered in Section 15.8.2, “Monitoring Databases from the Command Line” .

Table 15.6. Global Database Cache Information

Table Header	Description

Table Header	Description
Hits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
Tries	The total number of database accesses since server startup.
Hit Ratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
Pages Read In	The number of pages read from disk into the cache.
Pages Written Out	The number of pages written from the cache back to disk.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **Pages Written Out** in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts, the better.

15.7.2. Monitoring the Directory Server from the Command Line

The Directory Server's current activities can be monitored using LDAP tools such as **ldapsearch**, with the following characteristics:

- Search with the attribute filter **objectClass=***.
- Use the search base **cn=monitor**; the monitoring attributes for the server are found in the **cn=monitor** entry.
- Use the search scope **base**.

For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s base -b "cn=monitor" "
(objectclass=*)"
```

The monitoring attributes for the Directory Server are found in the **cn=monitor** entry. For information on searching the Directory Server, see Section 10.3, “Using ldapsearch”.

Table 15.7. Server Monitoring Attributes

Attribute	Description
version	Identifies the directory's current version number.
threads	The current number of active threads used for handling requests. Additional threads may be created by internal server tasks, such as replication or chaining.

Attribute	Description
connection:fd:opentime:opsinitiated:opscompleted:binddn:[rw]	Provides the following summary information for each open connection (only available if you bind to the directory as Directory Manager): *fd* – The file descriptor used for this connection. *opentime* – The time this connection was opened. *opsinitiated* – The number of operations initiated by this connection. *opscompleted* – The number of operations completed. *binddn* – The distinguished name used by this connection to connect to the directory. *rw* – The field shown if the connection is blocked for read or write. By default, this information is available to Directory Manager. However, the ACI associated with this information can be edited to allow others to access the information.
currentconnections	Identifies the number of connections currently in service by the directory.
totalconnections	Identifies the number of connections handled by the directory since it started.
dtablesize	Shows the number of file descriptors available to the directory. Each connection requires one file descriptor: one for every open index, one for log file management, and one for **ns-slapd** itself. Essentially, this value shows how many additional concurrent connections can be serviced by the directory. For more information on file descriptors, see the operating system documentation.
readwaiters	Identifies the number of threads waiting to read data from a client.
opsinitiated	Identifies the number of operations the server has initiated since it started.
opscompleted	Identifies the number of operations the server has completed since it started.
entriessent	Identifies the number of entries sent to clients since the server started.
bytessent	Identifies the number of bytes sent to clients since the server started.
currenttime	Identifies the time when this snapshot of the server was taken. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
starttime	Identifies the time when the server started. The time is displayed in Greenwich Mean Time (GMT) in UTC format.
nbackends	Identifies the number of back ends (databases) the server services.
backendmonitordn	Identifies the DN of each directory database.

15.8. MONITORING DATABASE ACTIVITY

The database's current activities can be monitored through Directory Server Console or from the command line.

Many performance counters in Directory Server use 32-bit numbers. However, under heavy loads, these 32-bit counter numbers can wrap too quickly. The ***nsslapd-counters*** attribute enabled the Directory Server to use 64-bit values for counter numbers, even on 32-bit systems. This enables long term statistics gathering for high-traffic systems.

Some of the counters for Directory Server database attributes monitored by server use 64-bit integers, even on 32-bit systems:

- entrycachehits
- entrycachetries
- currententrycachesize
- maxentrycachesize

The counters which use 64-bit integers are not configurable.

NOTE

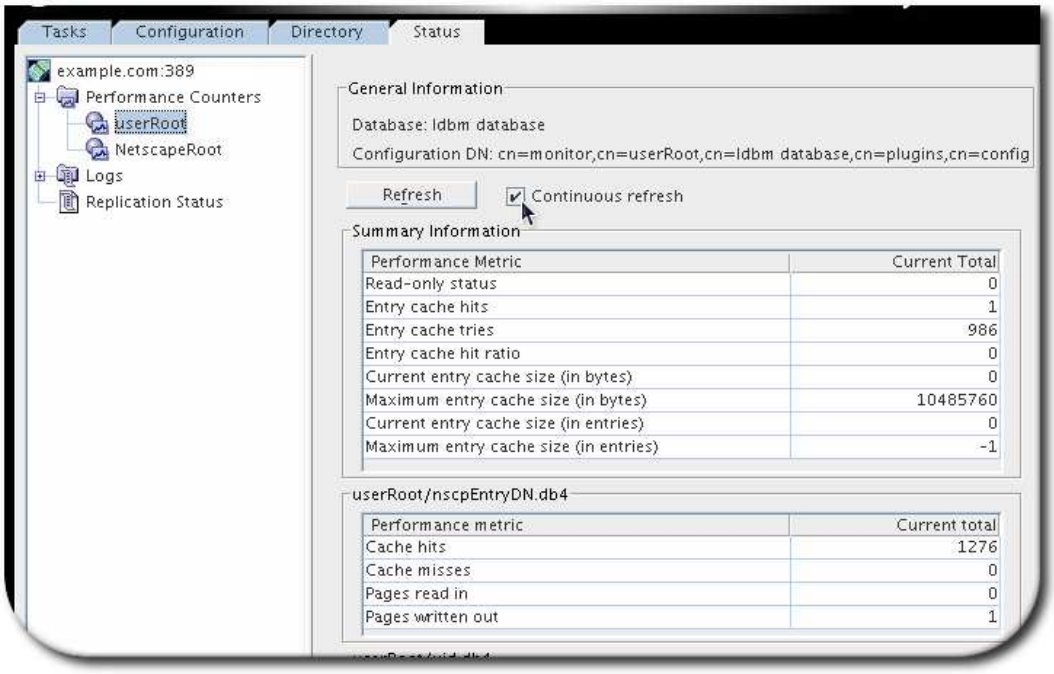
Tips for tuning the entry and database caches to improve server performance are in the *Tuning Red Hat Directory Server Performance*.

15.8.1. Monitoring Database Activity from the Directory Server Console

To monitor the database's activities:

1. In the Directory Server Console, select the **Status** tab.
2. In the navigation tree, expand the **Performance Counters** folder, and select the database to monitor.

   The tab displays current information about database activity. If the server is currently not running, this tab will not provide performance monitoring information.



3. Click **Refresh** to refresh the currently displayed information. For the directory to continuously update the displayed information, select the **Continuous** check box, and then click **Refresh**.

Table 15.8. General Information (Database)

Field	Description

Field	Description
Database	Identifies the type of database being monitored.
Configuration DN	Identifies the distinguished name that must be used as a search base to obtain these results using the **ldapsearch** command-line utility.

The **Summary Information** section shows the cumulative information for all of the databases being monitored and some cache-related configuration settings which are applied to all databases.

Table 15.9. Summary Information

Performance Metric	Current Total
Read-Only Status	Shows whether the database is currently in read-only mode. The database is in read-only mode when the ***nsslapd-readonly*** attribute is set to **on**.
Entry Cache Hits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
Entry Cache Tries	The total number of entry cache lookups since the directory was last started. That is, the total number of entries requested since server startup.
Entry Cache Hit Ratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever an operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the size of the entry cache by increasing the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
Current Entry Cache Size (in Bytes)	The total size of directory entries currently present in the entry cache.
Maximum Entry Cache Size (in Bytes)	The size of the entry cache maintained by the directory. This value is managed by the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. This is set in the **Memory available for cache** field in the database settings in the Directory Server Console.
Current Entry Cache Size (in Entries)	The number of directory entries currently present in the entry cache.
Maximum Entry Cache Size (in Entries)	*DEPRECATED.* The maximum number of directory entries that can be maintained in the entry cache. Do not attempt to manage the cache size by setting a maximum number of allowed entries. This can make it difficult for the host to allocate RAM effectively. Manage the cache size by setting the amount of RAM available to the cache, using the ***nsslapd-cachememsize*** attribute.

There are many different databases listed for the database monitoring page, by default, because databases are maintained for both entries and indexed attributes. All databases, though, have the same kind of cache information monitored in the counters.

Table 15.10. Database Cache Information

Performance Metric	Current Total
Hits	The number of times the database cache successfully supplied a requested page.
Tries	The number of times the database cache was asked for a page.
Hit Ratio	The ratio of database cache hits to database cache tries. The closer this value is to 100%, the better. Whenever a directory operation attempts to find a portion of the database that is not present in the database cache, the directory has to perform a disk access to obtain the appropriate database page. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory performance drops. To improve this ratio, increase the amount of data that the directory maintains in the database cache by increasing the value of the ***nsslapd-dbcachesize*** attribute. This is the **Maximum Cache Size** database setting in the Directory Server Console.
Pages Read In	The number of pages read from disk into the database cache.
Pages Written Out	The number of pages written from the cache back to disk. A database page is written to disk whenever a read-write page has been modified and then subsequently deleted from the cache. Pages are deleted from the database cache when the cache is full and a directory operation requires a database page that is not currently stored in cache.
Read-Only Page Evicts	The number of read-only pages discarded from the cache to make room for new pages.
Read-Write Page Evicts	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **Pages Written Out** in that these are discarded read-write pages that have not been modified.

Table 15.11. Database File-Specific

Performance Metric	Current Total
Cache Hits	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
Cache Misses	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
Pages Read In	The number of pages brought to the cache from this file.
Pages Written Out	The number of pages for this file written from cache to disk.

15.8.2. Monitoring Databases from the Command Line

A database's current activities can be monitored using LDAP tools such as **ldapsearch**. The search targets the monitoring subtree of the LDBM database entry, **cn=monitor,cn=***database_name***,cn=ldbm database,cn=plugins,cn=config**. This contains all of the monitoring attributes for the that specific database instance.

For example:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s base -b
"cn=monitor,cn=database_name,cn=ldbm database,cn=plugins,cn=config" "(objectclass=*)"
```

Table 15.12. Database Monitoring Attributes

Attribute	Description
database	Identifies the type of database currently being monitored.
readonly	Indicates whether the database is in read-only mode; **0** means that the server is not in read-only mode, **1** means that it is in read-only mode.
entrycachehits	The total number of successful entry cache lookups. That is, the total number of times the server could process a search request by obtaining data from the cache rather than by going to disk.
entrycachetries	The total number of entry cache lookups since the directory was last started. That is, the total number of search operations performed against the server since server startup.
entrycachehitratio	Ratio that indicates the number of entry cache tries to successful entry cache lookups. This number is based on the total lookups and hits since the directory was last started. The closer this value is to 100%, the better. Whenever a search operation attempts to find an entry that is not present in the entry cache, the directory has to perform a disk access to obtain the entry. Thus, as this ratio drops towards zero, the number of disk accesses increases, and directory search performance drops. To improve this ratio, increase the size of the entry cache by increasing the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
currententrycachesize	The total size, in bytes, of directory entries currently present in the entry cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.
maxentrycachesize	The maximum size, in bytes, of directory entries that can be maintained in the entry cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-cachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database. In the Directory Server Console, this is set in the **Memory available for cache** field in the database settings.

Attribute	Description
dbcachehits	The number of times the server could process a request by obtaining data from the cache rather than by going to the disk.
dbcachetries	The total number of database accesses since server startup.
dbcachehitratio	The ratio of cache tries to successful cache hits. The closer this number is to 100%, the better.
dbcachepagein	The number of pages read from disk into the cache.
dbcachepageout	The number of pages written from the cache back to disk.
dbcacheroevict	The number of read-only pages discarded from the cache to make room for new pages. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
dbcacherwevict	The number of read-write pages discarded from the cache to make room for new pages. This value differs from **Pages Written Out** in that these are discarded read-write pages that have not been modified. Pages discarded from the cache have to be written to disk, possibly affecting server performance. The lower the number of page evicts the better.
dbfilename-*number*	The name of the file. *number* provides a sequential integer identifier (starting at 0) for the file. All associated statistics for the file are given this same numerical identifier.
dbfilecachehit-*number*	The number of times that a search result resulted in a cache hit on this specific file. That is, a client performs a search that requires data from this file, and the directory obtains the required data from the cache.
dbfilecachemiss-*number*	The number of times that a search result failed to hit the cache on this specific file. That is, a search that required data from this file was performed, and the required data could not be found in the cache.
dbfilepagein-*number*	The number of pages brought to the cache from this file.
dbfilepageout-*number*	The number of pages for this file written from cache to disk.
currentdncachesize	The total size, in bytes, of DNs currently present in the DN cache. To increase the size of the entries which can be present in the DN cache, increase the value of the ***nsslapd-dncachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database.
maxdncachesize	The maximum size, in bytes, of DNs that can be maintained in the DN cache. To increase the size of the entries which can be present in the cache, increase the value of the ***nsslapd-dncachememsize*** attribute in the **cn=***database_name*, **cn=ldbm database,cn=plugins,cn=config** entry for the database.
currentdncachecount	The number of DNs currently present in the DN cache.

15.9. MONITORING DATABASE LINK ACTIVITY

It is possible to monitor the activity of database links from the command line using the **ldapsearch** command-line utility to return the monitoring attributes that are required. The monitoring attributes are stored in the **cn=monitor,cn=***database_link_name*, **cn=chaining database,cn=plugins,cn=config**.

For example, the **ldapsearch** command-line utility can be used to retrieve the number of add operations received by a particular database link. For example, this command monitors a database link called **DBLink1**:

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s sub -b
"cn=monitor,cn=DBLink1,cn=chaining database,cn=plugins,cn=config" "(objectclass=*)" nsAddCount
```

Table 15.13, "Database Link Monitoring Attributes" lists the database link monitoring attributes which can be monitored.

Table 15.13. Database Link Monitoring Attributes

Attribute Name	Description
nsAddCount	The number of add operations received.
nsDeleteCount	The number of delete operations received.
nsModifyCount	The number of modify operations received.
nsRenameCount	The number of rename operations received.
nsSearchBaseCount	The number of base-level searches received.
nsSearchOneLevelCount	The number of one-level searches received.
nsSearchSubtreeCount	The number of subtree searches received.
nsAbandonCount	The number of abandon operations received.
nsBindCount	The number of bind request received.
nsUnbindCount	The number of unbinds received.
nsCompareCount	The number of compare operations received.
nsOperationConnectionCount	The number of open connections for normal operations.
nsBindConnectionCount	The number of open connections for bind operations.

15.10. ENABLING AND DISABLING COUNTERS

The ***nsslapd-counters*** attribute enabled counters to run. However, running counters can affect performance, so it also possible to turn off counters. If counters are off, they all have a value of zero (0).

By default, counters are already enabled. To enable or disable performance counters, use **ldapmodify**:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=config
changetype: modify
replace: nsslapd-counters
nsslapd-counters: off
```

NOTE

Regardless of the whether the Directory Server is run on a 32-bit or 64-bit platform, the server itself records many counter values using 64-bit integers. For high-traffic sites, 32-bit counters turn over too quickly, which leads to quirky ratios and counts. Using 64-bit integers (on all platforms) improves performance and enables administrators to gather long-term statistics accurately.

CHAPTER 16. MONITORING DIRECTORY SERVER USING SNMP

The server and database activity monitoring log setup described in Chapter 15, *Monitoring Server and Database Activity* is specific to Directory Server. You can also monitor your Directory Server using Simple Network Management Protocol (SNMP), which is a management protocol used for monitoring network activity which can be used to monitor a wide range of devices in real time.

Directory Server can be monitored with SNMP through an AgentX subagent. SNMP monitoring collects useful information about the Directory Server, such as bind information, operations performed on the server, and cache information. The Directory Server SNMP subagent supports SNMP traps to send notifications about changes in the running state of your server instances.

16.1. ABOUT SNMP

SNMP has become interoperable on account of its widespread popularity. It is this interoperability, combined with the fact that SNMP can take on numerous jobs specific to a whole range of different device classes, that make SNMP the ideal standard mechanism for global network control and monitoring. SNMP allows network administrators to unify all network monitoring activities, with Directory Server monitoring part of the broader picture.

SNMP is used to exchange data about network activity. With SNMP, data travels between a managed device and a network management application (NMS) where users remotely manage the network. A managed device is anything that runs SNMP, such as hosts, routers, and your Directory Server. An NMS is usually a powerful workstation with one or more network management applications installed. A network management application graphically shows information about managed devices, which device is up or down, which and how many error messages were received, and so on.

Information is transferred between the NMS and the managed device through the use of two types of agents: the subagent and the *master agent*. The subagent gathers information about the managed device and passes the information to the master agent. Directory Server has a subagent. The master agent exchanges information between the various subagents and the NMS. The master agent usually runs on the same host machine as the subagents it talks to, although it can run on a remote machine.

Values for SNMP attributes, otherwise known as variables, that can be queried are kept on the managed device and reported to the NMS as necessary. Each variable is known as a *managed object*, which is anything the agent can access and send to the NMS. All managed objects are defined in a management information base (MIB), which is a database with a tree-like hierarchy. The top level of the hierarchy contains the most general information about the network. Each branch underneath is more specific and deals with separate network areas.

SNMP exchanges network information in the form of protocol data units (PDUs). PDUs contain information about variables stored on the managed device. These variables, also known as managed objects, have values and titles that are reported to the NMS as necessary. Communication between an NMS and a managed device takes place either by the NMS sending updates or requesting information or by the managed object sending a notice or warning, called a *trap*, when a server shuts down or starts up.

16.2. CONFIGURING THE MASTER AGENT

To use the subagent, you must have a master agent that supports AgentX. A common agent is Net-SNMP master agent, which may be available through your operating system vendor or can be downloaded from the Net-SNMP website, http://www.net-snmp.org.

The SNMP subagent included with Directory Server uses the AgentX protocol to communicate with the SNMP master agent running on your system. You must make sure that you enable AgentX support on your master agent. For Net-SNMP, add a line containing **agentx master** in the master agent's **snmpd.conf** file. For more details on configuring the master agent for AgentX support, see the Net-SNMP website, http://www.net-snmp.org.

16.3. CONFIGURING THE SUBAGENT

The Directory Server SNMP subagent is installed as a service on the Red Hat Enterprise Linux host machine, the same as the Directory Server and Admin Server processes.

16.3.1. Creating the Subagent Configuration File

The subagent is configured in a **.conf** configuration file. This file must be created manually. It can be named and located wherever you want, but it is strongly recommended that you use the default location. The default location is **/etc/dirsrv/config/ldap-agent.conf**. This file, as installed, is a template for the SNMP subagent service. The **ldap-agent.conf** file should be modified to fit the specific network environment.

This configuration file specifies the connection information for the master agent, logfile location, and which Directory Server instances to monitor.

16.3.1.1. agentx-master

The **agentx-master** setting tells the subagent how to communicate with the SNMP master agent. If this setting is not specified, the subagent tries to communicate the master agent through the Unix domain socket **/var/agentx/master**. This is also where the Net-SNMP master agent listens for AgentX communications by default. If you configured your master agent to listen on a different Unix domain socket, you must use the agentx-master setting for your subagent to communicate with your master agent by setting the new path for the **agentx-master** parameter. For example:

```
agentx-master /var/snmp/agentx
```

Make sure that the user as whom you are running the subagent has the appropriate permissions to write to this socket.

If the master agent is listening for AgentX communications on a TCP port, the **agentx-master** setting has the host name and port number for the master agent. For example:

```
agentx-master localhost:705
```

16.3.1.2. agent-logdir

The **agent-logdir** setting specifies the directory where the subagent will write its logfile. For example:

```
agent-logdir /var/log
```

If this parameter is not specified, the agent will write its logfile to the same location as your subagent configuration file. The logfile will be named **ldap-agent.log**.

The default log directory as specified in the **ldap-agent.conf** template is **/var/log/dirsrv/**. This is the same directory where instance log files are stored, but the SNMP aubagent log files are not instance-specific. It is recommended that you use the default location.

Make sure that the user as whom your subagent is running has write permission to this directory.

16.3.1.3. server

The server setting specifies a Directory Server instance that you want to monitor. You must use one server setting for each Directory Server instance. The subagent requires at least one server setting to be specified in its configuration file. The server setting should be set to the name of the Directory Server instance you would like to monitor. For example:

```
server slapd-phonebook
```

To monitor multiple Directory Server instances, an additional **server** parameter in the subagent configuration file for each instance.

```
server slapd-phonebook
server slapd-example
server slapd-directory
```

16.3.2. Starting the Subagent

Once your master agent is running and you have created your subagent configuration file, start the subagent.

NOTE

The Directory Server does not have to be started for the subagent to be started.

The subagent is started and runs as a separate process than the Red Hat Directory Server or Admin Server processes. To start or stop the subagent, use the **service** system tools.

```
service dirsrv-snmp {start|stop|restart}
```

The status of the subagent can also be checked with the **service** tool.

```
service dirsrv-snmp status
dirsrv-snmp is stopped
```

16.3.3. Testing the Subagent

To test your subagent, use any SNMP client tools to query the master agent. Net-SNMP contains simple command-line utilities such as **snmpwalk** and **snmpget**. In order for these tools to use variable names for queries, configure them to load the Directory Server's MIB file. The Directory Server's MIB file, **redhat-directory.mib**, is located in **/usr/share/dirsrv/mibs** on Red Hat Enterprise Linux 6 (64-bit). There are some additional common required MIB files in this **mibs** directory if you do not already have them with your MIB tools.

The MIB file is not needed for the subagent to operate; it is only required for any SNMP client application to use variable names instead of numeric OIDs to see the monitored information provided by the subagent.

Each monitored server instance uses its port number as an index to identify that particular Directory Server instance. For example, querying for the **dsEntityName.389** SNMP variable returns the variable value for a server running on port 389, assuming that instance exists and is being monitored by the subagent.

For details on configuring and using the Net-SNMP command-line tools, check out the Net-SNMP website, http://www.net-snmp.org.

16.4. CONFIGURING SNMP TRAPS

An SNMP trap is essentially a threshold which triggers a notification if it is encountered by the monitored server. To use traps, the master agent must be configured to accept traps and do something with them. For example, a trap can trigger an email notification for an administrator of the Directory Server instance stops.

The subagent is only responsible for sending the traps to the master agent. The master agent and a trap handler must be configured according to the documentation for the SNMP master agent you are using.

Traps are accompanied by information from the **Entity Table**, which contains information specific to the Directory Server instance, such as its name and version number. The **Entity Table** is described in Section 16.6.3, “Entity Table”. This means that the action the master agent takes when it receives a trap is flexible, such as sending an email to an email address defined in the ***dsEntityContact*** variable for one instance while sending a notification to a pager number in the ***dsEntityContact*** variable for another instance.

There are two traps supported by the subagent:

- *DirectoryServerDown*. This trap is generated whenever the subagent detects the Directory Server is potentially not running. This trap will be sent with the Directory Server instance description, version, physical location, and contact information, which are detailed in the ***dsEntityDescr***, ***dsEntityVers***, ***dsEntityLocation***, and ***dsEntityContact*** variables.

- *DirectoryServerStart*. This trap is generated whenever the subagent detects that the Directory Server has started or restarted. This trap will be sent with the Directory Server instance description, version, physical location, and contact information, which are detailed in the ***dsEntityDescr***, ***dsEntityVers***, ***dsEntityLocation***, and ***dsEntityContact*** variables.

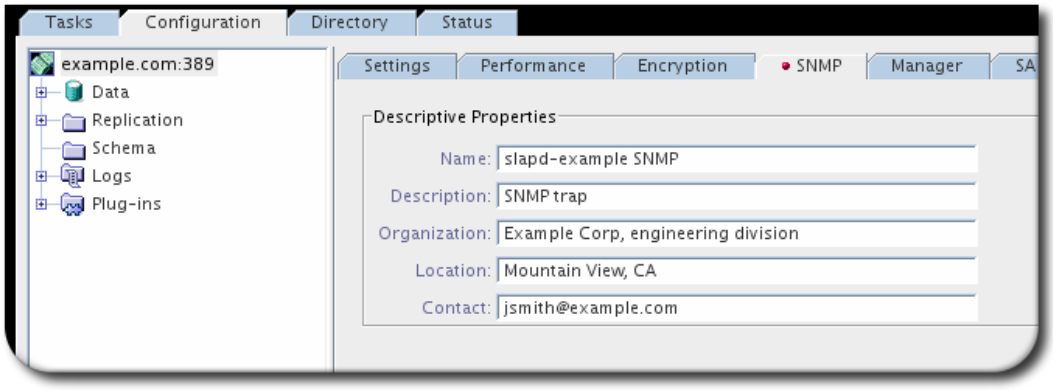
16.5. CONFIGURING THE DIRECTORY SERVER FOR SNMP

By default, the Directory Server is ready to be monitored using SNMP as soon as the subagent is configured. However, there are some useful variables in the Directory Server instances which can be configured to help identify the Directory Server instance with SNMP. To configure these SNMP settings from the Directory Server Console:

1. Select the **Configuration** tab, and then select the topmost entry in the navigation tree in the left pane.

2. Select the **SNMP** tab in the main window.

3. Fill in the information about the SNMP descriptors so that it is easy to identify the Directory Server instance in Net-SNMP.



- A unique name and description for the instance.
- The company or organization to which the directory instance belongs.
- The physical location of the directory instance or the organization which manages the instance.
- The email address or contact number for the person who maintains the Directory Server instance.

4. Click **Save**.

16.6. USING THE MANAGEMENT INFORMATION BASE

The Directory Server's MIB is a file called **redhat-directory.mib**. This MIB contains definitions for variables pertaining to network management for the directory. These variables are known as managed objects. Using the directory MIB and Net-SNMP, you can monitor your directory like all other managed devices on your network. For more information on using the MIB, see Section 16.3.3, "Testing the Subagent".

The client tools need to load the Directory Server MIB to use the variable names listed in the following sections.

Using the directory MIB enables administrators to use SNMP to see administrative information about the directory and monitor the server in real-time. The directory MIB is broken into four distinct tables of managed objects:

- Section 16.6.1, "Operations Table"
- Section 16.6.2, "Entries Table"
- Section 16.6.3, "Entity Table"
- Section 16.6.4, "Interaction Table"

NOTE

All of the Directory Server attributes monitored by SNMP use 64-bit integers for the counters, even on 32-bit systems.

16.6.1. Operations Table

The **Operations Table** provides statistical information about Directory Server access, operations, and errors. Table 16.1, "Operations Table: Managed Objects and Descriptions" describes the managed objects stored in the **Operations Table** of the **redhat-directory.mib** file.

Table 16.1. Operations Table: Managed Objects and Descriptions

Managed Object	Description
dsAnonymousBinds	The number of anonymous binds to the directory since server startup.
dsUnauthBinds	The number of unauthenticated binds to the directory since server startup.
dsSimpleAuthBinds	The number of binds to the directory that were established using a simple authentication method (such as password protection) since server startup.

Managed Object	Description
dsStrongAuthBinds	The number of binds to the directory that were established using a strong authentication method (such as SSL or a SASL mechanism like Kerberos) since server startup.
dsBindSecurityErrors	The number of bind requests that have been rejected by the directory due to authentication failures or invalid credentials since server startup.
dsInOps	The number of operations forwarded to this directory from another directory since server startup.
dsReadOps	The number of read operations serviced by this directory since application start. The value of this object will always be **0** because LDAP implements read operations indirectly using the search operation.
dsCompareOps	The number of compare operations serviced by this directory since server startup.
dsAddEntryOps	The number of add operations serviced by this directory since server startup.
dsRemoveEntryOps	The number of delete operations serviced by this directory since server startup.
dsModifyEntryOps	The number of modify operations serviced by this directory since server startup.
dsModifyRDNOps	The number of modify RDN operations serviced by this directory since server startup.
dsListOps	The number of list operations serviced by this directory since server startup. The value of this object will always be **0** because LDAP implements list operations indirectly using the search operation.
dsSearchOps	The total number of search operations serviced by this directory since server startup.
dsOneLevelSearchOps	The number of one-level search operations serviced by this directory since server startup.
dsWholeSubtreeSearchOps	The number of whole subtree search operations serviced by this directory since server startup.
dsReferrals	The number of referrals returned by this directory in response to client requests since server startup.
dsSecurityErrors	The number of operations forwarded to this directory that did not meet security requirements.
dsErrors	The number of requests that could not be serviced due to errors (other than security or referral errors). Errors include name errors, update errors, attribute errors, and service errors. Partially serviced requests will not be counted as an error.

16.6.2. Entries Table

The **Entries Table** provides information about the contents of the directory entries. Table 16.2, "Entries Table: Managed Objects and Descriptions" describes the managed objects stored in the **Entries Table** in the **redhat-directory.mib** file.

Table 16.2. Entries Table: Managed Objects and Descriptions

Managed Object	Description
dsMasterEntries	The number of directory entries for which this directory contains the master entry. The value of this object will always be **0** (as no updates are currently performed).
dsCopyEntries	The number of directory entries for which this directory contains a copy.The value of this object will always be **0** (as no updates are currently performed).
dsCacheEntries	The number of entries cached in the directory.
dsCacheHits	The number of operations serviced from the locally held cache since application startup.
dsSlaveHits	The number of operations that were serviced from locally held replications (shadow entries). The value of this object will always be **0**.

16.6.3. Entity Table

The **Entity Table** contains identifying information about the Directory Server instance. The values for the **Entity Table** are set in the Directory Server Console, as described in Section 16.5, "Configuring the Directory Server for SNMP".

Table 16.3, "Entity Table: Managed Objects and Descriptions" describes the managed objects stored in the **Entity Table** of the **redhat-directory.mib** file.

Table 16.3. Entity Table: Managed Objects and Descriptions

Managed Object	Description
dsEntityDescr	The description set for the Directory Server instance.
dsEntityVers	The Directory Server version number of the Directory Server instance.
dsEntityOrg	The organization responsible for the Directory Server instance.
dsEntityLocation	The physical location of the Directory Server instance.
dsEntityContact	The name and contact information for the person responsible for the Directory Server instance.
dsEntityName	The name of the Directory Server instance.

16.6.4. Interaction Table

NOTE

The **Interaction Table** is *not* supported by the subagent. The subagent can query the table, but it will not ever be updated with valid data.

Table 16.4, "Interaction Table: Managed Objects and Descriptions" describes the managed objects stored in the **Interaction Table** of the **redhat-directory.mib** file.

Table 16.4. Interaction Table: Managed Objects and Descriptions

Managed Object	Description

Managed Object	Description
dsIntTable	Details, in each row of the table, related to the history of the interaction of the monitored Directory Servers with their respective peer Directory Servers.
dsIntEntry	The entry containing interaction details of a Directory Server with a peer Directory Server.
dsIntIndex	Part of the unique key, together with **applIndex**, to identify the conceptual row which contains useful information on the (attempted) interaction between the Directory Server (referred to by **applIndex**) and a peer Directory Server.
dsName	The distinguished name (DN) of the peer Directory Server to which this entry belongs.
dsTimeOfCreation	The value of **sysUpTime** when this row was created. If the entry was created before the network management subsystem was initialized, this object will contain a value of zero.
dsTimeOfLastAttempt	The value of **sysUpTime** when the last attempt was made to contact this Directory Server. If the last attempt was made before the network management subsystem was initialized, this object will contain a value of zero.
dsTimeOfLastSuccess	The value of **sysUpTime** when the last attempt made to contact this Directory Server was successful. This entry will have a value of zero if there have been no successful attempts or if the last successful attempt was made before the network management subsystem was initialized.
dsFailuresSinceLastSuccess	The number of failures since the last time an attempt to contact this Directory Server was successful. If there has been no successful attempts, this counter will contain the number of failures since this entry was created.
dsFailures	Cumulative failures since the creation of this entry.
dsSuccesses	Cumulative successes since the creation of this entry.
dsURL	The URL of the Directory Server application.

CHAPTER 17. PLANNING FOR DISASTER

Part of running a Directory Server deployment efficiently is planning for that worst case scenario. This chapter covers general principles for drafting a disaster recovery plan and highlights features in Directory Server that can be used to aide in disaster recovery.

Disaster recovery is a way of planning and implementing a smooth transition from one operating environment to another environment whenever there is some sort of catastrophic failure. A disaster recovery plan for Directory Server may be part of a larger business continuity plan or it could be a standalone plan specifically for an interruption in directory services.

NOTE

This chapter covers very general concepts for disaster recovery.

Disaster recovery can be a very complex and detail-specific thing. Consider using a professional service to design, maintain, and test any disaster recovery plan for sensitive or mission-critical services, like Red Hat Directory Server.

17.1. IDENTIFYING POTENTIAL SCENARIOS

The first step is identifying what potential issues you may encounter, what services will be affected, and what responses you should take. In the *Directory Server Deployment Guide*, administrators made a site survey of their existing and proposed infrastructure to determine what kind of directory to design. Do something similar for disaster planning; as in Table 17.1, “Disaster Scenarios and Responses”, identify where your data infrastructure is, determine what the affect of losing that component is, and look at potential ideal responses.

Table 17.1. Disaster Scenarios and Responses

Scenario	Effects on Infrastructure	Ideal Response
Data corruption	Through software or hardware failure (or through a malicious attack), the data at one site or on one server could be corrupted. If that corrupted server is a supplier in multi-master replication, then the corruption can quickly be propagated throughout the deployment.	An isolated server should be available with access to the most recent backup of uncorrupted data. When a problem is detected, replication can be suspended on the regular infrastructure, and this server can be brought online to reinitialize the suppliers with good data.
Natural disasters and other mass events	Natural disasters can take an entire office or data center offline, even through something as simple as a long-term power outage.	Directory operations can be transferred to a mirrored site at another physical location, with the same data.
Server or machine loss	A single machine could fail.	Another machine, with the same data, can assume the lost machine's place.

17.2. DEFINING THE TYPE OF ROLLOVER

Disaster recovery, as the introduction says, is the process for transitioning from one system to another system with as little interruption of service as possible. That's called a *rollover*, and there are three different ways of doing a rollover:

- A *hot* rollover means that the infrastructure is completely mirrored at another site and that the backup site is always up and current with the primary site. This requires only a few adjustments to switch operations from the primary to the backup.
- A *warm* rollover means that all of the elements for the backup site are in place (adequate network connections, all required applications and hardware) but the system is not actively running or necessarily configured. This can require some extra time to configure the machines and get the system running.
- A *cold* rollover means that a site is available but there are few resources immediately available to set it up.

The obvious difference in the types of rollover is the time and expense necessary to set up the backup site. Hot and warm sites have higher initial expenditures to set up and run.

A mix of rollover types can be used, depending on the specific disaster scenario being planned. For example, a rollover plan for the loss of a single server could use a hot rollover easily and relatively cheaply by creating and

keeping a virtual machine copy of the Directory Server instance which can be brought online within minutes. It would not even require keeping the virtual machine in a separate facility or network. On the other hand, a cold rollover could be planned for the loss of an entire data center or office.

Match the rollover process to the severity of the disaster scenario, your budget and available resources, and the likelihood of encountering problems.

17.3. IDENTIFYING USEFUL DIRECTORY SERVER FEATURES FOR DISASTER RECOVERY

The hardest part of a recovery is not the hardware; it is getting a reliable copy of the data in the server. There are three Directory Server features that are excellent tools for preparing data copies for disaster recovery:

- Multi-master replication, chaining, backing up databases, and monitoring the server with a named pipe script.
- Chaining
- Backing up databases regularly

Additionally, monitoring the server with a named pipe script and with other Directory Server performance counters can be effective at catching and quickly responding to specific, critical events.

17.3.1. Multi-Master Replication for Disaster Recovery

Multi-master replication is the best defense against losing a single server and, possibly, even an entire office or department. While a small number of servers are data masters, multiple servers all hold the same data — potentially dozens of masters and hubs in a single replication environment. This keeps information accessible to clients even if multiple servers go offline.

Replication can be used to copy over data to servers and bring replacements online more quickly.

NOTE

To protect against data corruption being propagated through replication, schedule a lag in replication to a backup server or to another server cluster. This way, replication can be stopped before the corrupt data are replicated over to the backup site.

Replication configuration also allows write operations to be referred to failover servers if the primary supplier is inaccessible. This means that write operations can proceed as normal from the client perspective, even when servers go offline.

> **Example 17.1. Scenarios for Multi-Master Replication**
>
> Replication is a versatile tool for disaster recovery in several scenarios:
>
> - For a single server failure, all of the data stored on that instance is both accessible and retrievable from other servers.
> - For the loss of an entire office or colocation facility, servers can be mirrored at an entirely different physical location (which is aided by Directory Server's wide area replication performance). With minimal effort, traffic can be redirected to the replicated site without having to bring new servers online.

Configuring replication is covered in Chapter 11, *Managing Replication*.

17.3.2. Chaining Databases for Disaster Recovery

Chaining is a configuration where a client sends a request to one server and it automatically forwards that request to another server to process. There can be multiple servers configured in the database link (or chain) to allow for automatic failover if one server is not available.

> **Example 17.2. Scenarios for Chaining**
>
> When chaining is combined with a list of failover servers, client traffic can be automatically redirected from a single server (or even group of servers) when they are offline. This does not help in recovery, but it helps manage the transition from primary to backup servers.

Chaining databases is covered in Section 2.3, “Creating and Maintaining Database Links” .

17.3.3. Backing up Directory Data for Disaster Recovery

The most useful tool for disaster recovery is to do frequent backups of a directory instance. Archives can be stored on physical media, at different locations than the primary data center or on-site at a cold backup location. Because both backup and restore operations can be done through either shell or perl scripts (such as **db2bak.pl**) backups can be automated to run regularly through simple cron jobs.

```
0 7 * * 1 /usr/lib[64]/dirsrv/slapd-example/db2bak.pl
```

NOTE

The **db2bak.pl** Perl script backs up the directory data without having to stop the server first.

Backing up both directory databases and the directory configuration (**dse.ldif** file) are covered in Section 4.3, "Backing up and Restoring Data".

17.3.4. Using a Named Pipe Script for Disaster Recovery

Named pipe scripts can be used to do two very good things: identify specific error messages and work with plug-ins or other scripts to respond to events. For certain kinds of failures and recoveries, this could be used to essentially automate a response, such as rerouting client traffic or bringing a virtual machine online.

Using named pipe scripts is described in Section 15.5, "Replacing Log Files with a Named Pipe" .

17.4. DEFINING THE RECOVERY PROCESS

There are a lot of tools that can help with disaster recovery, but an effective recovery process circles back to having a well-defined plan of what to do in every scenario. Two things, at least, need to be clearly identified:

- What signals a disaster? Some things are obvious (a massive power outage, network loss, or fire), but other situations need to be defined. For example, what signals that a backup server needs to be brought online?
- Who responds to a disaster and how? Once a disaster situation occurs, who has the responsibility to act? How are they notified of the event? What are they expected to do?

After a disaster plan is written for each scenario, then test the plan regularly (at least yearly). As the server configuration changes, be sure to update the plan so it reflects the current infrastructure.

17.5. BASIC EXAMPLE: PERFORMING A RECOVERY

An administrator, John Smith, has to create a disaster recovery plan for his directory deployment. Example Corp. has three physical offices, in San Francisco, Dallas, and Arlington. Each site has 10 servers which replicate to each other locally, and then one server at each site replicates to another server at the other two sites.

Each site has business-critical customer data stored in its directory, as well as human resources data. Several external applications require access to the data to perform operations like billing.

John Smith's first step is to perform a site survey. He is looking for three things: what his directory usage is (clients that access it and traffic loads across the sites), what his current assets are, and what assets he may need to acquire. This is much like the initial site survey he performed when deploying Red Hat Directory Server.

His next step is identifying potential disaster scenarios. Two of the three sites are highly vulnerable to natural disasters (San Francisco and Dallas). All three sites could face normal interruptions, like outages for power or Internet access. Additionally, since each site masters its own local data, each site is vulnerable to losing a server instance or machine.

John Smith then breaks his disaster recovery plan into three parts:

- Plan A covers losing a single instance of Directory Server
- Plan B covers some kind of data corruption or attack
- Plan C covers losing an entire office

For plans A and B, John Smith decides to use a hot recovery to immediately switch functionality from a single instance to the backup. Each server is backed up daily, using a cron job, and then the archive is copied over and restored on a virtual machine. The virtual machine is kept on a different subnet, but can be switched over immediately if its peer ever does offline. John Smith uses simple SNMP traps to track each Directory Server instance's availability.

Plan C is more extensive. Along with replication between sites and the local backups, he decides to mail a physical

copy of each site's backup, for every local instance, once a week to the other two colocation facilities. He also keep a spare server with adequate Internet access and software licenses to restore an entire site, using virtual machines, one of the other different colocation facilities. He designates the Arlington site as the primary recovery location because that is where most of the IT staff is located, then San Francisco and last Dallas, based on the distribution of personnel. For every event, the IT administrator at all three sites will be notified, and the manager assumes the responsibilities of setting up the virtual machines, restoring the Directory Server instances from the physical backups, and rerouting client traffic.

John Smith schedules to review and update the plan quarterly to account for any new hardware or application changes. Once a year, all three sites have to run through the procedure of recovering and deploying the other two sites, according to the procedures in Disaster Plan C.

APPENDIX A. USING LDAP CLIENT TOOLS

Red Hat Directory Server 9.0 uses the LDAP tools (such as **ldapsearch** and **ldapmodify**) supplied with OpenLDAP. The OpenLDAP tool options are described in the OpenLDAP manpages at http://www.openldap.org/software/man.cgi.

This appendix gives some common usage scenarios and examples for using these LDAP tools.

More extensive examples for using **ldapsearch** are given in Chapter 10, *Finding Directory Entries*. More examples for using **ldapmodify** and **ldapdelete** are given in Section 3.2, "Managing Entries from the Command Line".

A.1. ENVIRONMENT VARIABLES USED WITH LDAP CLIENT TOOLS

Some information related to running LDAP client tools can be set through environment variables. This allows certain operation conditions (like SSL/TLS settings) to be set once and then applied consistently to every operation.

NOTE

The SSL/TLS parameters can be set as either an environment variable or within the OpenLDAP configuration, meaning set in **/etc/openldap/ldap.conf** or the **$HOME/[.]ldaprc** profiles.

Table A.1. LDAP Tools Environment Variables

Environment Variable	ldap.conf Parameter	Description
LDAP_BASEDN	none	Sets the default base DN for **ldapsearch** to use. This is equivalent to the **-b** argument and allows that argument to be skipped.
LDAPTLS_CACERTDIR	TLS_CACERTDIR	Gives the directory where the NSS security databases (**cert8.db** and **key3.db**) are located. For example, **/etc/dirsrv/slapd-*instance_name***.
LDAPTLS_CERT	TLS_CERT	Gives the nickname for the server certificate in the **cert8.db** database. For example, **Server-Cert**.
LDAPTLS_KEY	TLS_KEY	Gives the password and, optionally, the token name which stores the key, in the format *[token_name:]password*. The default token name (which is assumed) is **internal**. For example, **internal:secret** or **secret**.

A.2. USING SSL/TLS AND START TLS WITH LDAP CLIENT TOOLS

There are three things that must be configured for SSL/TLS connections with LDAP command-line tools:

- The appropriate environment variables must be set before running the command.
- The appropriate arguments must be passed with the command to identify the server to which to connect. This requires either **-H** to specify an LDAP or LDAPI URL or **-h** and **-p** to give the fully-qualified domain name and port of the server.

  NOTE

  The fully-qualified domain name is *always* required with the **-h** option. This prevents man-in-the-middle attacks.

- SSL/TLS must be specified. This can be done by invoking Start TLS with the **-Z** or **-ZZ** (force Start TLS) options. Start TLS is described in Section 7.5, "Command-Line Functions for Start TLS". For example:

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub
-x -ZZ "(objectclass=*)"
```

Alternatively, this can be done using the LDAPS protocol with **-H** or using the secure LDAPS port with the **-p** option. Although using the **ldaps** protocol is supported, it is deprecated. The recommended method is to use Start TLS.

When using SSL/TLS with LDAP command-line tools for client connections, the appropriate TLS environment variables (Section A.1, "Environment Variables Used with LDAP Client Tools") must be set in order to access the required security databases and certificates.

To set up *system-wide* SSL/TLS configuration for LDAP tools, edit the **ldap.conf** file. To edit *per-user* SSL/TLS configuration for LDAP tools, edit the **$HOME/.ldaprc** profile for the specific user. Whichever file is edited, the same configuration parameters need to be set.

1. Open the **ldap.conf** file or **$HOME/.ldaprc** profile. For example:

```
vim /etc/openldap/ldap.conf
```

NOTE

These parameters can also be set as environment variables. See Table A.1, "LDAP Tools Environment Variables" for the variable names.

2. Add a line to define the security databases directory location. This is required.

```
TLS_CACERTDIR /etc/dirsrv/slapd-instance_name
```

3. Optionally, add lines for the client certificate name and token database password for the Directory Server's NSS security databases. This allows certificate-based client authentication.

```
TLS_CERT Server-Cert
TLS_KEY internal:secret
```

Once the security databases parameters are set, then SSL connections can be invoked with the LDAP command-line tools. For example, using **-ZZ** opens a Start TLS connection and forces the use of TLS or the operation fails:

```
ldapsearch -D "cn=directory manager" -W -p 389 -h server.example.com -b "dc=example,dc=com" -s sub -x -ZZ "
(objectclass=*)"
```

The **-x** allows simple (user name/password) binds. Alternatively, use the **-Y EXTERNAL** option to indicate that an authentication method other than SASL is being used. The **-Y EXTERNAL** argument can be used with client authentication:

```
ldapsearch -H ldaps://server.example.com:636 -b "dc=example,dc=com" -Y EXTERNAL "givenname=Richard"
```

A.3. USING SASL WITH LDAP CLIENT TOOLS

Directory Server uses SASL for authentication and network security, particularly for environments which are using Kerberos to implement single sign-on. Directory Server allows user to use SASL to authenticate and bind to the server and then to encrypt (secure) the network connection to the server.

SASL authentication and security include LDAP tools like **ldapsearch** and **ldapmodify**. For example:

```
ldapsearch -O noplain,minssf=1,maxbufsize=512 -Y GSSAPI -U "dn:uid=jsmith,ou=people,dc=example,dc=com" -
R EXAMPLE.COM ...
```

The SASL-related LDAP tool parameters are listed in Table A.2, "LDAP Client Tool SASL Parameters".

NOTE

SASL proxy authorization is not supported in Directory Server; therefore, Directory Server ignores any SASL **authzid** value supplied by the client.

Two primary pieces of information are required to use SASL with Directory Server:

- The authentication method, in this example GSS-API
- The user as whom you are authenticating (the authorization ID)

Other information, such as the Kerberos realm, can also be passed with the command.

When a client connects to Directory Server using SASL, the Directory Server takes the identity offered as the SASL ***authid*** and maps that entry back to an entry in the Directory Server. If the ***authid*** is defined as a DN (as in **authid=dn:***DN*), this is done simply by matching the DN. It is also possible to use a user name or a part of a DN, and these can be mapped to the directory entry using SASL identity mappings.

Table A.2. LDAP Client Tool SASL Parameters

Option	Description	Allowed Values
-O	*Optional.* Sets the security properties for the connection.	All mechanisms: • *noplain* – Do not permit mechanisms susceptible to simple passive attack. • *noanonymous* – Do not permit mechanisms that allow anonymous access. • *minssf* – Require a minimum security strength; this option needs a numeric value specifying bits of encryption. A value of **- 1** means integrity is provided without privacy. • *maxssf* – Require a maximum security strength; this option needs a numeric value specifying bits of encryption. A value of **- 1** means integrity is provided without privacy. CRAM-MD mechanism only: • *noactive* – Do not permit mechanisms susceptible to active attacks. • *nodict* – Do not permit mechanisms susceptible to passive dictionary attacks. • *forwardsec* – Require forward secrecy. • *passcred* – Attempt to pass client credentials. • *maxbufsize* – Set the maximum receive buffer size the client will accept when using integrity or privacy settings.
-R	Gives the Kerberos realm.	Depends on the mechanism.
-U	Gives the ID used to authenticate to the server.	• *UID.* For example, **msmith**. • *u: uid.* For example, **u: msmith**. • *dn: dn_value.* For example, **dn: uid=msmith,ou=People,o =example.com**.

Option	Description	Allowed Values
-Y	Sets the SASL authentication mechanism to use.	• GSSAPI • CRAM-MD5 • DIGEST-MD5 • EXTERNAL • PLAIN

A.4. RUNNING EXTENDED OPERATIONS

Red Hat Directory Server supports a variety of extended operations, especially extended search operations. An extended operation passes an additional operation (such as a get effective rights search or server-side sort) along with the LDAP operation. Likewise, LDAP clients have the potential to support a number of extended operations.

The OpenLDAP LDAP tools support extended operations in two ways. All client tools (**ldapmodify**, **ldapsearch**, and the others) use either the **-e** or **-E** options to send an extended operation. The **-e** argument can be used with any OpenLDAP client tool and sends general instructions about the operation, like how to handle password policies. The **-E** is used only with **ldapsearch**es and passes more useful controls like GER searches, sort and page information, and information for other, not-explicitly-support extended operations.

Additionally, OpenLDAP has another tool, **ldapexop**, which is used exclusively to perform extended search operations, the same as running **ldapsearch -E**.

The format of an extended operation with **ldapsearch** is generally:

```
-E extended_operation_type=operation_parameters
```

When an extended operation is explicitly handled by the OpenLDAP tools, then the *extended_operation_type* can be an alias, like **deref** for a dereference search or **sss** for server-side sorting. A supported extended operation has formatted output. Other extended operations, like GER searches, are passed using their OID rather than an alias, and then the *extended_operation_type* is the OID. For those unsupported operations the tool does not recognize the response from the server, so the output is unformatted.

For example, the **pg** extended operation type formats the results in simple pages:

```
ldapsearch -x -D "cn=Directory Manager" -W -b "ou=Engineers,ou=People,dc=example,dc=com" -E pg=3 "
(objectclass=*)" cn

dn: uid=jsmith,ou=Engineers,ou=People,dc=example,dc=com
  cn: John Smith

dn: uid=bjensen,ou=Engineers,ou=People,dc=example,dc=com
  cn: Barbara Jensen

dn: uid=hmartin,ou=Engineers,ou=People,dc=example,dc=com
  cn: Henry Martin

Results are sorted.
next page size (3): 5
```

The same operation with **ldapexop** can be run using only the OID of the simple paged results operation and the operation's settings (3 results per page):

```
ldapexop 1.2.840.113556.1.4.319=3
```

However, **ldapexop** does not accept the same range of search parameters that **ldapsearch** does, making it less flexible.

A.5. ADDING ENTRIES

There are two ways to add entries using LDAP client tools:

- **ldapadd**

- **ldapmodify -a**

Both **ldapmodify -a** and **ldapadd** are exactly the same; the only difference is that with **ldapmodify**, the **-a** option explicitly states that it is an add operation. With **ldapadd**, the **-a** option is implied.

The example using **ldamodify -a** is given in Section 3.2.4.1, "Adding Entries Using ldapmodify". To add entries using **ldapadd**, use the same options and format as **ldamodify -a**:

```
ldapadd -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -f new.ldif


ldapadd -D "cn=directory manager" -w secret -p 389 -h server.example.com -x

dn: uid=jsmith,ou=people,dc=example,dc=com
objectclass: inetOrgPerson
objectclass: person
objectclass: top
uid: jsmith
sn: Smith
givenname: John
mail: jsmith@example.com
cn: John Smith
```

A.6. COMPARING ENTRIES

ldapcompare checks entries to see if the specified entry or entries contain an attribute of a specific value. For example, this checks to see if an entry has an ***sn*** value of Smith:

```
ldapcompare -D "cn=directory manager" -w secret -p 389 -h server.example.com -x sn:smith
uid=bjensen,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=bjensen,ou=people,dc=example,dc=com"
compare FALSE

ldapcompare -D "cn=directory manager" -w secret -p 389 -h server.example.com -x sn:smith
uid=jsmith,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=jsmith,ou=people,dc=example,dc=com"
compare TRUE
```

The compare attribute can be specified in one of three ways:

- A single *attribute:value* statement passed in the command line directly

  ```
  sn:Smith
  ```

- A single *attribute::base64value* statement passed in the command line directly, for attributes like ***jpegPhoto*** or to verify certificates or CRLs

  ```
  jpegPhoto:dkdkPDKCDdko0eiofk==
  ```

- An *attribute:file* statement that points to a file containing a list of comparison values for the attribute, and the script iterates through the list

  ```
  postalCode:/tmp/codes.txt
  ```

The compare operation itself has to be run against a specific entry or group of entries. A single entry DN can be passed through the command line, or a list of DNs to be compared can be given using the **-f** option.

Example A.1. Comparing One Attribute Value to One Entry

Both the attribute-value comparison and the DN are passed with the script.

```
ldapcompare -D "cn=directory manager" -w secret -p 389 -h server.example.com -x sn:smith
uid=jsmith,ou=people,dc=example,dc=com
comparing type: "sn" value: "smith" in entry "uid=jsmith,ou=people,dc=example,dc=com"
compare TRUE
```

Example A.2. Comparing a List Attribute Values from a File

First, create a file of possible ***sn*** values.

```
jensen
johnson
johannson
jackson
jorgenson
```

Then, create a list of entries to compare the values to.

```
uid=jen200,ou=people,dc=example,dc=com
uid=dsj,ou=people,dc=example,dc=com
uid=matthewjms,ou=people,dc=example,dc=com
uid=john1234,ou=people,dc=example,dc=com
uid=jack.son.1990,ou=people,dc=example,dc=com
```

Then run the script.

```
ldapcompare -D "cn=directory manager" -w secret -p 389 -h server.example.com -x sn:/tmp/surnames.txt -f
/tmp/names.txt
comparing type: "sn" value: "jensen" in entry "uid=jen200,ou=people,dc=example,dc=com"
compare TRUE
```

A.7. CHANGING PASSWORDS

The **ldappasswd** command can either set a new user-defined password or generate a new password for an account. Table A.3, "Password Operation-Related Parameters for ldappasswd" lists the most important parameters for setting passwords through the command line. Other settings (for bind information, connection information, or other command settings) may be required and are listed in the OpenLDAP manpages.

```
ldappasswd -x -D bind_dn -w password -p server_port -h server_hostname [-A | -a oldPassword] [-S | -s
newPassword] [user]
```

IMPORTANT

Password change operations must be run over a secure connection, such as SSL/TLS, Start TLS, or SASL. For information on how to configure SSL/TLS for LDAP clients, see Section A.2, "Using SSL/TLS and Start TLS with LDAP Client Tools".

Table A.3. Password Operation-Related Parameters for ldappasswd

Option	Description
-A	Prompts for the original password, which is being changed.
-a	Gives the old password, which is being changed.
-n	Tells the server not to set a new password. This is mainly used with the **-v** option (which increases the verbosity of the output) or the **-d** option (which sets the debug level) by testing the output without actually performing a password change operation.
-S	Prompts for the new password.
-s	Sets the new password.
user	Gives the DN of the user entry for which to change the password.

Example A.3. Directory Manager Changing a User's Password Over SSL

The Directory Manager changes the password of the user **uid=tuser1,ou=People,dc=example,dc=com** to new_password over SSL.

```
ldappasswd -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -s new_password
"uid=tuser1,ou=People,dc=example,dc=com"
```

Example A.4. Directory Manager Generating a User's Password

The Directory Manager generates the password of the user **uid=tuser2,ou=People,dc=example,dc=com** over SSL.

```
ldappasswd -D "cn=directory manager" -w secret -p 389 -h server.example.com -x
"uid=tuser2,ou=People,dc=example,dc=com"
```

Example A.5. User Changing His Own Password

A user, **tuser3**, changes the password from **old_newpassword** to **new_password** over SSL.

```
ldappasswd -p 389 -h server.example.com -x -D "uid=tuser3,ou=People,dc=example,dc=com" -W -a
old_password -s new_password
```

Example A.6. User Authenticating with DIGEST_MD5 and Changing His Password

A user, jsmith, authenticates with GSS-API and changes the password to *new_password*.

```
ldappasswd -p 389 -h server.example.com -O noplain,minssf=1,maxbufsize=512 -Y GSSAPI -U
"dn:uid=jsmith,ou=people,dc=example,dc=com" -R EXAMPLE.COM -W -s new_password
```

Example A.7. User Already Authenticated by Kerberos Prompts for a New Password

A user, who has already authenticated by Kerberos, prompts for the new password. This is not performed over SSL.

```
ldappasswd -p 389 -h server.example.com -O noplain,minssf=1,maxbufsize=512 -I
```

A.8. GENERATING LDAP URLS

LDAP URLs are used in a variety of different configuration areas and operations: referrals and chaining, replication, synchronization, ACIs, and indexing, as a starting list. Constructing accurate LDAP URLs is critical, because incorrect URLs may connect to the wrong server or simply cause operations to fail. Additionally, all OpenLDAP tools allow the **-H** option to pass an LDAP URL instead of other connection information (like the host name, port, subtree, and search base).

NOTE

LDAP URLs are described in Appendix C, *LDAP URLs*.

The **ldapurl** command manages URL in two ways:

- Deconstruct a given LDAP URL into its constituent element
- Construct a new, valid LDAP URL from given elements

The parameters for working with URLs are listed in Table A.4, "ldapurl Parameters"; the full list of parameters are in the OpenLDAP manpages.

Table A.4. ldapurl Parameters

Option	Description

Option	Description
For Deconstructing a URL	
-H "*URL*"	Passes the LDAP URL to break down into elements.
For Constructing a URL	
-a *attributes*	Gives a comma-separated attributes that are specifically returned in search results.
-b *base*	Sets the search base or subtree for the URL.
-f *filter*	Sets the search filter to use.
-h *hostname*	Gives the Directory Server's host name.
-p *port*	Gives the Directory Server's port.
-S ldap|ldaps|ldapi	Gives the protocol to use to connect, such as **ldap**, **ldaps**, or **ldapi**.
-s *scope*	Gives the search scope.

Example A.8. Deconstructing an LDAP URL

ldapurl uses the **-H** option to feed in an existing LDAP URL, and the tool returns the elements of the URL in a neat list:

```
ldapurl -H "ldap://:389/dc=example,dc=com?cn,sn?sub?(objectclass=inetorgperson)"
scheme: ldap
port: 389
dn: dc=example,dc=com
selector: cn
selector: sn
scope: sub
filter: (objectclass=inetorgperson)
```

Example A.9. Constructing an LDAP URL

The most useful application of **ldapurl** is to construct a valid LDAP URL manually. The Directory Server Console has tools to develop valid URLs for areas like ACIs and referrals, but very complex configurations or scripted operations may require administrators to manually construct the URL. Using **ldapurl** ensures that the URL is valid.

ldapurl accepts the normal connection parameters of all LDAP client tools and additional **ldapsearch** arguments for search base, scope, and attributes, but this tool never connects to a Directory Server instance, so it does not require any bind information. It accepts the connection and search settings and feeds them in as elements to the URL.

```
ldapurl -a cn,sn -b dc=example,dc=com -s sub -f "(objectclass=inetorgperson)"

ldap://:389/dc=example,dc=com?cn,sn?sub?(objectclass=inetorgperson)
```

APPENDIX B. LDAP DATA INTERCHANGE FORMAT

Red Hat Directory Server (Directory Server) uses the LDAP Data Interchange Format (LDIF) to describe a directory and directory entries in text format. LDIF is commonly used to build the initial directory database or to add large numbers of entries to the directory all at once. In addition, LDIF is also used to describe changes to directory entries. For this reason, most of Directory Server's command-line utilities rely on LDIF for either input or output.

Because LDIF is a text file format, LDIF files can be created using virtually any language. All directory data is stored using the UTF-8 encoding of Unicode. Therefore, the LDIF files created must also be UTF-8 encoded.

For information on using LDIF to modify directory entries, see Chapter 3, *Creating Directory Entries*.

B.1. ABOUT THE LDIF FILE FORMAT

LDIF consists of one or more directory entries separated by a blank line. Each LDIF entry consists of an optional entry ID, a required distinguished name, one or more object classes, and multiple attribute definitions.

The LDIF format is defined in RFC 2849, *The LDAP Data Interchange Format (LDIF)* . Directory Server is compliant with this standard.

The basic form of a directory entry represented in LDIF is as follows:

```
dn: distinguished_name
objectClass: object_class
objectClass: object_class
...
 attribute_type[;subtype]:attribute_value
...
```

- Every LDIF entry must have a DN and at least one object class definition.
- Include any attributes required by the object classes defined for the entry.
- All other attributes and object classes are optional.
- Object classes and attributes can be specified in any order.
- The space after the colon is optional.

For information on standard object classes and attributes, see the *Red Hat Directory Server 9 Configuration, Command, and File Reference*.

Table B.1, "LDIF Fields" describes the LDIF fields shown in the previous definition.

Table B.1. LDIF Fields

Field	Definition
[*id*]	*Optional*. A positive decimal number representing the entry ID. The database creation tools generate this ID automatically. Never add or edit this value yourself.
dn: *distinguished_name*	Specifies the distinguished name for the entry.
objectClass: *object_class*	Specifies an object class to use with this entry. The object class identifies the types of attributes, or schema, allowed and required for the entry. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of standard object classes and Chapter 8, *Managing the Directory Schema* for information on customizing the schema.
attribute_type	Specifies a descriptive attribute to use with the entry. The attribute should be defined either in the schema. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of standard attributes and Chapter 8, *Managing the Directory Schema* for information on customizing the schema.

Field	Definition
[*subtype*]	*Optional.* Specifies subtype, language, binary, or pronunciation. Use this tag to identify the language in which the corresponding attribute value is expressed or whether the attribute value is binary or a pronunciation of an attribute value. For information on attribute subtypes, see Section 3.1.3.5, "Adding an Attribute Subtype". For a complete list of the supported subtypes tags, see Table D.2, "Supported Language Subtypes".
attribute_value	Specifies the attribute value to be used with the attribute type.

NOTE

The LDIF syntax for representing a change to an entry in the directory is different from the syntax described in Table B.1, "LDIF Fields". For information on using LDIF to modify directory entries, see Chapter 3, *Creating Directory Entries*.

B.2. CONTINUING LINES IN LDIF

In LDIF files, a line can be broken and continued (called *folded*) by indenting the continued portion of the line by exactly one space. For example, the following two statements are identical:

```
dn: cn=Jake Lupinski,dc=example,dc=com

dn: cn=Jake Lup
 inski,dc=exa
 mple,dc=com
```

It is not required to break and continue LDIF lines. However, doing so may improve the readability of the LDIF file. The usual convention is that an LDIF file does not contain more than 78 columns of text.

B.3. REPRESENTING BINARY DATA

Binary data, such as a JPEG image, is represented in LDIF using one of two methods, standard LDIF notation or base-64 encoding.

B.3.1. Standard LDIF Notation

Standard LDIF notation uses the lesser than (<) symbol to indicate that the data are binary. For example:

```
jpegphoto: < file:/path/to/photo
```

With this standard notation, it is not necessary to specify the **ldapmodify -b** parameter. However, standard notation requires that the following line be added to the beginning of the LDIF file or the LDIF update statements:

```
version: 1
```

For example:

```
ldapmodify -x -D userDN -W

version: 1
dn: cn=Barney Fife,ou=People,dc=example,dc=com
changetype: modify
add: usercertificate
usercertificate;binary: < file: BarneysCert
```

B.3.2. Base-64 Encoding

Binary data can be converted to base-64, which can be used in LDIF files, for a variety of data, from images to SSL certificates. Base 64-encoded data are identified by using the **::** symbol. For example:

```
jpegPhoto::encoded_data
```

In addition to binary data, other values that must be base-64 encoded include the following:

- Any value that begins with a colon (:) or a space.
- Any value that contains non-ASCII data, including new lines.

Use the **ldif** command-line utility with the **-b** parameter to convert binary data to LDIF format:

```
ldif -b attribute_name
```

attribute_name is the name of the attribute to which the binary data is supplied. The binary data is read from standard input and the results are written to standard output. Thus, use redirection operators to select input and output files.

The **ldif** command-line utility will take any input and format it with the correct line continuation and appropriate attribute information. The **ldif** utility also assesses whether the input requires base-64 encoding. For example:

```
ldif -b jpegPhoto <  mark.jpg > out.ldif
```

This example takes a binary file containing a JPEG-formatted image and converts it into LDIF format for the attribute ***jpegPhoto***. The output is saved to **out.ldif**.

The **-b** option specifies that the **ldif** utility should interpret the entire input as a single binary value. If **-b** is not present, each line is considered to be a separate input value.

B.4. SPECIFYING DIRECTORY ENTRIES USING LDIF

Many types of entries can be stored in the directory. This section concentrates on three of the most common types of entries used in a directory: domain, organizational unit, and organizational person entries.

The object classes defined for an entry are what indicate whether the entry represents a domain or domain component, an organizational unit, an organizational person, or some other type of entry. For a complete list of the object classes that can be used by default in the directory and a list of the most commonly used attributes, see the *Red Hat Directory Server 9 Configuration, Command, and File Reference* .

B.4.1. Specifying Domain Entries

Directories often have at least one domain entry. Typically this is the first, or topmost, entry in the directory. The domain entry often corresponds to the DNS host and domain name for your directory. For example, if the Directory Server host is called **ldap.example.com**, then the domain entry for the directory is probably named **dc=ldap,dc=example,dc=com** or simply **dc=example,dc=com**.

The LDIF entry used to define a domain appears as follows:

```
dn: distinguished_name
objectClass: top
objectClass: domain
dc: domain_component_name
 list_of_optional_attributes
...
```

The following is a sample domain entry in LDIF format:

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
description: Fictional example company
```

Each element of the LDIF-formatted domain entry is defined in Table B.2, “LDIF Elements in Domain Entries” .

Table B.2. LDIF Elements in Domain Entries

LDIF Element	Description
dn: *distinguished_name*	*Required.* Specifies the distinguished name for the entry.

LDIF Element	Description
objectClass: top	*Required.* Specifies the **top** object class.
objectClass: domain	Specifies the **domain** object class. This line defines the entry as a domain or domain component. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes that can be used with this object class.
dc: *domain_component*	Attribute that specifies the domain's name. The server is typically configured during the initial setup to have a suffix or naming context in the form **dc=***hostname*,**dc=***domain*,**dc=***toplevel*. For example, **dc=ldap,dc=example,dc=com**. The domain entry should use the leftmost ***dc*** value, such as **dc: ldap**. If the suffix were **dc=example,dc=com**, the ***dc*** value is **dc: example**. Do not create the entry for **dn: dc=com** unless the server has been configured to use that suffix.
list_of_attributes	Specifies the list of optional attributes to maintain for the entry. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes that can be used with this object class.

B.4.2. Specifying Organizational Unit Entries

Organizational unit entries are often used to represent major branch points, or subdirectories, in the directory tree. They correspond to major, reasonably static entities within the enterprise, such as a subtree that contains people or a subtree that contains groups.

The organizational unit attribute that is contained in the entry may also represent a major organization within the company, such as marketing or engineering. However, this style is discouraged. Red Hat strongly encourages using a flat directory tree.

There is usually more than one organizational unit, or branch point, within a directory tree.

The LDIF that defines an organizational unit entry must appear as follows:

```
dn: distinguished_name
objectClass: top
objectClass: organizationalUnit
ou: organizational_unit_name
 list_of_optional_attributes
...
```

The following is a sample organizational unit entry in LDIF format:

```
dn: ou=people,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: people
description: Fictional example organizational unit
```

Table B.3, "LDIF Elements in Organizational Unit Entries" defines each element of the LDIF-formatted organizational unit entry.

Table B.3. LDIF Elements in Organizational Unit Entries

LDIF Element	Description
dn: *distinguished_name*	Specifies the distinguished name for the entry. A DN is required. If there is a comma in the DN, the comma must be escaped with a backslash (\), such as **dn: ou=people,dc=example,dc=com**.

LDIF Element	Description
objectClass: top	*Required.* Specifies the **top** object class.
objectClass: organizationalUnit	Specifies the **organizationalUnit** object class. This line defines the entry as an **organizational unit**. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes available for this object class.
ou: *organizational_unit_name*	Attribute that specifies the organizational unit's name.
list_of_attributes	Specifies the list of optional attributes to maintain for the entry. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes available for this object class.

B.4.3. Specifying Organizational Person Entries

The majority of the entries in the directory represent organizational people.

In LDIF, the definition of an organizational person is as follows:

```
dn: distinguished_name
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: common_name
sn: surname
 list_of_optional_attributes
```

The following is an example organizational person entry in LDIF format:

```
dn: uid=bjensen,ou=people,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
cn: Babs Jensen
sn: Jensen
givenname: Babs
uid: bjensen
ou: people
description: Fictional example person
telephoneNumber: 555-5557
userPassword: {SSHA}dkfljlk34r2kljdsfk9
```

Table B.4, "LDIF Elements in Person Entries" defines each aspect of the LDIF person entry.

Table B.4. LDIF Elements in Person Entries

LDIF Element	Description
dn: *distinguished_name*	*Required.* Specifies the distinguished name for the entry. For example, **dn: uid=bjensen,ou=people,dc=example,dc=com**. If there is a comma in the DN, the comma must be escaped with a backslash (\).
objectClass: top	*Required.* Specifies the **top** object class.
objectClass: person	Specifies the **person** object class. This object class specification should be included because many LDAP clients require it during search operations for a person or an organizational person.

LDIF Element	Description
objectClass: organizationalPerson	Specifies the **organizationalPerson** object class. This object class specification should be included because some LDAP clients require it during search operations for an organizational person.
objectClass: inetOrgPerson	Specifies the **inetOrgPerson** object class. The **inetOrgPerson** object class is recommended for the creation of an organizational person entry because this object class includes the widest range of attributes. The ***uid*** attribute is required by this object class, and entries that contain this object class are named based on the value of the ***uid*** attribute. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes available for this object class.
cn: *common_name*	Specifies the person's common name, which is the full name commonly used by the person. For example, **cn: Bill Anderson**. At least one common name is required.
sn: *surname*	Specifies the person's surname, or last name. For example, **sn: Anderson**. A surname is required.
list_of_attributes	Specifies the list of optional attributes to maintain for the entry. See the *Red Hat Directory Server 9 Configuration, Command, and File Reference* for a list of the attributes available for this object class.

B.5. DEFINING DIRECTORIES USING LDIF

The contents of an entire directory can be defined using LDIF. Using LDIF is an efficient method of directory creation when there are many entries to add to the directory.

To create a directory using LDIF:

1. Create an ASCII file containing the entries to add in LDIF format.

   Make sure each entry is separated from the next by an empty line. Use just one line between entries, and make sure the first line of the file is not be blank, or else the **ldapmodify** utility will exit. For more information, see Section B.4, “Specifying Directory Entries Using LDIF” .

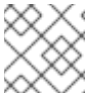
2. Begin each file with the topmost, or root, entry in the database.

   The root entry must represent the suffix or sub-suffix contained by the database. For example, if the database has the suffix **dc=example,dc=com**, the first entry in the directory must be **dn: dc=example,dc=com**.

   For information on suffixes, see the "Suffix" parameter described in the *Directory Server Configuration and Command-Line Tool Reference*.

3. Make sure that an entry representing a branch point in the LDIF file is placed before the entries to create under that branch.

   For example, to place an entry in a people and a group subtree, create the branch point for those subtrees before creating entries within those subtrees.

   NOTE

   The LDIF file is read in order, so parent entries must be listed before the child entries.

4. Create the directory from the LDIF file using one of the following methods:

   - *Initializing the database through the Directory Server Console.* Use this method if there is a small database to import (less than 10,000 entries). See Section 4.1.4, “Importing a Database from the Console”.

WARNING

This method is destructive and will erase any existing data in the suffix.

- *ldif2db or ldif2db.pl command-line utility.* Use this method if there is a large database to import (more than 10,000 entries). See Section 4.1.6.1, "Importing Using the ldif2db Command-Line Script".
  - **ldif2db** cannot be used if the server is running.
  - **ldif2db.pl** can only be used if the server is running.

WARNING

This method is destructive and will erase any existing data in the suffix.

- *ldapmodify command-line utility with the -a parameter.* Use this method if a new subtree is being added to an existing database or there is existing data in the suffix which should not be deleted. Unlike the other methods for creating the directory from an LDIF file, Directory Server must be running before a subtree can be added using **ldapmodify**. See Section 3.2.4, "Adding and Modifying Entries Using ldapmodify".

Example B.1. LDIF File Example

This LDIF file contains one domain, two organizational units, and three organizational person entries:

```
dn: dc=example,dc=com
objectclass: top
objectclass: domain
dc: example
description: Fictional example domain

dn: ou=People,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: People
description: Fictional example organizational unit
tel: 555-5559

dn: cn=June Rossi,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: June Rossi
sn: Rossi
givenName: June
mail: rossi@example.com
userPassword: {sha}KDIE3AL9DK
ou: Accounting
ou: people
telephoneNumber: 2616
roomNumber: 220

dn: cn=Marc Chambers,ou=People,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Marc Chambers
sn: Chambers
givenname: Marc
```

```
mail: chambers@example.com
userPassword: {sha}jdl2alem87dlacz1
telephoneNumber: 2652
ou: Manufacturing
ou: People
roomNumber: 167

dn: cn=Robert Wong,ou=People,example.com Corp,dc=example,dc=com
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Robert Wong
cn: Bob Wong
sn: Wong
givenname: Robert
givenname: Bob
mail: bwong@example.com
userPassword: {sha}nn2msx761
telephoneNumber: 2881
roomNumber: 211
ou: Manufacturing
ou: people

dn: ou=Groups,dc=example,dc=com
objectclass: top
objectclass: organizationalUnit
ou: groups
description: Fictional example organizational unit
```

B.6. STORING INFORMATION IN MULTIPLE LANGUAGES

If the directory contains a single language, it is not necessary to do anything special to add a new entry to the directory. However, if an organization is multinational, it may be necessary to store information in multiple languages so that users in different locales can view directory information in their own language.

When information in the directory is represented in multiple languages, the server associates language tags with attribute values. When a new entry is added, the attribute values used in the RDN (relative distinguished name, the naming attribute) must be provided without any language codes.

Multiple languages can be stored for a single attribute. In this case, the attribute types are the same, but each value has a different language code.

For a list of the languages supported by Directory Server and their associated language tags, see Section D.2, “Supported Locales”.

NOTE

The language tag has no effect on how the string is stored within the directory. All object class and attribute strings are stored using UTF-8. The user is responsible for converting the data used in the LDIF to UTF-8. The **iconv** or **uconv** command provided by most operating systems can be used to convert data from the native characterset into UTF-8.

For example, Example Corporation has offices in the United States and France and wants employees to be able to view directory information in their native language. When adding directory entries, the directory administrator chooses to provide attribute values in both English and French. When adding a directory entry for a new employee, Babs Jensen, the administrator does the following:

1. The administrator creates a file, **street.txt**, with the French street address value:

   ```
   1 rue de l'Université
   ```

2. The file contents are then converted to UTF-8:

   ```
   iconv -t UTF-8 -o output.txt street.txt
   ```

3. The following LDIF entry is created using the UTF-8 value of the street address value for ***streetAddress;lang-fr***.

```
dn: uid=bjensen,ou=people,dc=example,dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
name: Babs Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
streetAddress: 1 University Street
streetAddress;lang-en: 1 University Street
streetAddress;lang-fr:: AasljdoaAJASI023909jaASJaonasd0ADS
preferredLanguage: fr
```

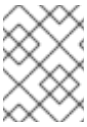
The double colons after the attribute name and subtype indicate that the value is binary base-64 encoded.

Users accessing this directory entry with an LDAP client with the preferred language set to English will see the address **1 University Street**. Users accessing the directory with an LDAP client with the preferred language set to French will see the address **1 rue de l'Université**.

APPENDIX C. LDAP URLS

LDAP URLs identify the Red Hat Directory Server instance, similarly to the way site URLs identify a specific website or web page. There are three common times when the LDAP URL of the Directory Server instance is used:

- The LDAP URL is used to identify the specific Directory Server instance when the Directory Server is accessed using a web-based client.
- LDAP URLs are used to configure Directory Server referrals.
- LDAP URLs are used to configure access control instructions.



NOTE

The LDAP URL format is described in RFC 4516, which is available at http://www.ietf.org/rfc/rfc4516.txt.

C.1. COMPONENTS OF AN LDAP URL

LDAP URLs have the following syntax:

```
ldap[s]://hostname:port/base_dn?attributes?scope?filter
```

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

The **ldap://** protocol is used to connect to LDAP servers over unsecured connections, and the **ldaps://** protocol is used to connect to LDAP servers over TLS/SSL connections. Table C.1, "LDAP URL Components" lists the components of an LDAP URL.

NOTE

The LDAP URL format is described in RFC 4516, which is available at http://www.ietf.org/rfc/rfc4516.txt.

Table C.1. LDAP URL Components

Component	Description
host name	Name (or IPv4 or IPv6 address) of the LDAP server. For example, **ldap.example.com** or **192.202.185.90**.
port	Port number of the LDAP server (for example, **696**). If no port is specified, the standard LDAP port (**389**) or LDAPS port (**636**) is used.
base_dn	Distinguished name (DN) of an entry in the directory. This DN identifies the entry that is the starting point of the search. If no base DN is specified, the search starts at the root of the directory tree.
attributes	The attributes to be returned. To specify more than one attribute, use commas to separate the attributes; for example, **cn,mail,telephoneNumber**. If no attributes are specified in the URL, all attributes are returned.

Component	Description
scope	The scope of the search, which can be one of these values: **base** retrieves information only about the distinguished name (*base_dn*) specified in the URL. **one** retrieves information about entries one level below the distinguished name (*base_dn*) specified in the URL. The base entry is not included in this scope. **sub** retrieves information about entries at all levels below the distinguished name (*base_dn*) specified in the URL. The base entry is included in this scope. If no scope is specified, the server performs a **base** search.
filter	Search filter to apply to entries within the specified scope of the search. If no filter is specified, the server uses the filter **(objectClass=*)**.

The attributes, scope, and filter components are identified by their positions in the URL. Even if no attributes are specified, the question marks still must be included to delimit that field.

For example, to specify a subtree search starting from **dc=example,dc=com** that returns all attributes for entries matching **(sn=Jensen)**, use the following LDAP URL:

```
ldap://ldap.example.com/dc=example,dc=com??sub?(sn=Jensen)
```

The two consecutive question marks, **??**, indicate that no attributes have been specified. Since no specific attributes are identified in the URL, all attributes are returned in the search.

C.2. ESCAPING UNSAFE CHARACTERS

Any *unsafe* characters in the URL need to be escaped, or substituted with a special sequence of characters.

For example, a space is an unsafe character that must be represented as **%20** within the URL. Thus, the distinguished name **o=example.com corporation** must be encoded as **o=example.com%20corporation**.

The following table lists the characters that are considered unsafe within URLs and provides the associated escape characters to use in place of the unsafe character:

Unsafe Character	Escape Characters
space	%20
<	%3c
>	%3e
"	%22
#	%23
%	%25
{	%7b
}	%7d
|	%7c
\	%5c

Unsafe Character	Escape Characters
^	%5e
~	%7e
[	%5b
]	%5d
`	%60

C.3. EXAMPLES OF LDAP URLS

NOTE

The LDAP URL format is described in RFC 4516, which is available at http://www.ietf.org/rfc/rfc4516.txt.

Example 1

The following LDAP URL specifies a base search for the entry with the distinguished name **dc=example,dc=com**.

```
ldap://ldap.example.com/dc=example,dc=com
```

- Because no port number is specified, the standard LDAP port number (**389**) is used.
- Because no attributes are specified, the search returns all attributes.
- Because no search scope is specified, the search is restricted to the base entry **dc=example,dc=com**.
- Because no filter is specified, the directory uses the default filter (**objectclass=***).

Example 2

The following LDAP URL retrieves the ***postalAddress*** attribute of the entry with the DN **dc=example,dc=com**:

```
ldap://ldap.example.com/dc=example,dc=com?postalAddress
```

- Because no search scope is specified, the search is restricted to the base entry **dc=example,dc=com**.
- Because no filter is specified, the directory uses the default filter (**objectclass=***).

Example 3

The following LDAP URL retrieves the ***cn***, ***mail***, and ***telephoneNumber*** attributes of the entry for Barbara Jensen:

```
ldap://ldap.example.com/cn=Barbara%20Jensen,dc=example,dc=com?cn,mail,telephoneNumber
```

- Because no search scope is specified, the search is restricted to the base entry **cn=Barbara Jensen,dc=example,dc=com**.
- Because no filter is specified, the directory uses the default filter **(objectclass=*)**.

Example 4

The following LDAP URL specifies a search for entries that have the surname **Jensen** and are at any level under **dc=example,dc=com**:

```
ldap://ldap.example.com/dc=example,dc=com??sub?(sn=Jensen)
```

- Because no attributes are specified, the search returns all attributes.
- Because the search scope is **sub**, the search encompasses the base entry **dc=example,dc=com** and entries at all levels under the base entry.

Example 5

The following LDAP URL specifies a search for the object class for all entries one level under **dc=example,dc=com**:

```
ldap://ldap.example.com/dc=example,dc=com?objectClass?one
```

- Because the search scope is **one**, the search encompasses all entries one level under the base entry **dc=example,dc=com**. The search scope does not include the base entry.
- Because no filter is specified, the directory uses the default filter (**objectclass=***).

NOTE

The syntax for LDAP URLs does not include any means for specifying credentials or passwords. Search requests initiated through LDAP URLs are unauthenticated, unless the LDAP client that supports LDAP URLs provides an authentication mechanism.

APPENDIX D. INTERNATIONALIZATION

Red Hat Directory Server allows users to store, manage, and search for entries and their associated attributes in a number of different languages. An internationalized directory can be an invaluable corporate resource, providing employees and business partners with immediate access to the information they need in languages they understand.

Directory Server supports all international charactersets by default because directory data is stored in UTF-8. Further, Directory Server can use specified matching rules and collation orders based on language preferences in search operations.

NOTE

ASCII characters are required for attribute and object class names.

D.1. ABOUT LOCALES

Directory Server provides support for multiple languages through the use of *locales*. A locale identifies language-specific information about how users of a specific region, culture, or custom expect data to be presented, including how data of a given language is interpreted and how data is to be sorted, or *collated*.

In addition, the locale information indicates what code page should be used to represent a given language. A code page is an internal table that the operating system uses to relate keyboard keys to character font screen displays.

More specifically, a locale defines four things:

- *Collation order*. The collation order provides language and cultural-specific information about how the characters of a given language are to be sorted. It identifies things like the sequence of the letters in the alphabet, how to compare letters with accents to letters without accents, and if there are any characters that can be ignored when comparing strings. The collation order also takes into account culture-specific information about a language, such as the direction in which the language is read (left to right, right to left, or up and down).

- *Character type*. The character type distinguishes alphabetic characters from numeric or other characters. For example, in some languages, the pipe (|) character is considered punctuation while in others it is considered alphabetic. In addition, it defines the mapping of upper-case to lower-case letters.

- *Monetary format*. The monetary format specifies the monetary symbol used by a specific region, whether the symbol goes before or after its value, and how monetary units are represented.

- *Time/date format*. The time and date format indicates the customary formatting for times and dates in the region. The time and date format indicates whether dates are customarily represented in the *mm/dd/yy* (month, day, year) or *dd/mm/yy* (day, month, year) format and specifies what the days of the week and month are in a given language. For example, the date January 10, 1996, is represented as **10.leden 1996** in Czechoslovakian and **10 janvier 1996** in French.

Because a locale describes cultural, customary, and regional differences in addition to mechanical language differences, the directory data can both be translated into the specific languages understood by users as well as be presented in a way that users in a given region expect.

D.2. SUPPORTED LOCALES

When performing directory operations that require that a locale be specified, such as a search operation, use a language tag or a collation order object identifier (OID).

A *language tag* is a string that begins with the two-character lowercase language code that identifies the language, as defined in ISO Standard 639. If necessary to distinguish regional differences in language, the language tag may also contain a two-character string for the country code, as defined in ISO Standard 3166. The language code and country code are separated by a hyphen. For example, the language tag used to identify the British English locale is **en-GB**.

An *object identifier* (OID) is a decimal number used to uniquely identify an object, such as an attribute or object class. The OIDs for searching or indexing an internationalized directory identify specific collation orders supported by the Directory Server. For example, the OID **2.16.840.1.113730.3.3.2.17.1** identifies the Finnish collation order.

When performing an international search in the directory, use either the language tag or the OID to identify the collation order to use. However, when setting up an international index, the OIDs must be used. For more information on indexing, see Chapter 9, *Managing Indexes*.

Table D.1, “Supported Locales” lists each locale supported by Directory Server and identifies the associated language tags and OIDs.

Table D.1. Supported Locales

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
Albanian	sq	2.16.840.1.113730.3.3.2.44.1
Arabic	ar	2.16.840.1.113730.3.3.2.1.1
Belorussian	be	2.16.840.1.113730.3.3.2.2.1
Bulgarian	bg	2.16.840.1.113730.3.3.2.3.1
Catalan	ca	2.16.840.1.113730.3.3.2.4.1
Chinese (Simplified)	zh	2.16.840.1.113730.3.3.2.49.1
Chinese (Traditional)	zh-TW	2.16.840.1.113730.3.3.2.50.1
Croatian	hr	2.16.840.1.113730.3.3.2.22.1
Czechoslovakian	cs	2.16.840.1.113730.3.3.2.5.1
Danish	da	2.16.840.1.113730.3.3.2.6.1
Dutch	nl or nl-NL	2.16.840.1.113730.3.3.2.33.1
Dutch (Belgian)	nl-BE	2.16.840.1.113730.3.3.2.34.1
English (US)	en or en-US	2.16.840.1.113730.3.3.2.11.1
Estonian	et	2.16.840.1.113730.3.3.2.16.1
Finnish	fi	2.16.840.1.113730.3.3.2.17.1
French	fr or fr-FR	2.16.840.1.113730.3.3.2.18.1
French (Belgian)	fr-BE	2.16.840.1.113730.3.3.2.19.1
French (Canadian)	fr-CA	2.16.840.1.113730.3.3.2.20.1
French (Swiss)	fr-CH	2.16.840.1.113730.3.3.2.21.1
German	de	2.16.840.1.113730.3.3.2.7.1
German (Austrian)	de-AT	2.16.840.1.113730.3.3.2.8.1
German (Swiss)	de-CH	2.16.840.1.113730.3.3.2.9.1
Greek	el	2.16.840.1.113730.3.3.2.10.1
Hebrew	iw	2.16.840.1.113730.3.3.2.27.1
Hungarian	hu	2.16.840.1.113730.3.3.2.23.1
Icelandic	is	2.16.840.1.113730.3.3.2.24.1
Italian	it	2.16.840.1.113730.3.3.2.25.1
Italian (Swiss)	it-CH	2.16.840.1.113730.3.3.2.26.1

Locale	Language Tag	Collation Order Object Identifiers (OIDs)
Japanese	ja	2.16.840.1.113730.3.3.2.28.1
Korean	ko	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish	lv	2.16.840.1.113730.3.3.2.31.1
Lithuanian	lt	2.16.840.1.113730.3.3.2.30.1
Macedonian	mk	2.16.840.1.113730.3.3.2.32.1
Norwegian	no	2.16.840.1.113730.3.3.2.35.1
Norwegian (Bokmul)	nb	2.16.840.1.113730.3.3.2.36.1
Norwegian (Nynorsk)	no-NO-NY	2.16.840.1.113730.3.3.2.37.1
Polish	pl	2.16.840.1.113730.3.3.2.38.1
Romanian	ro	2.16.840.1.113730.3.3.2.39.1
Russian	ru	2.16.840.1.113730.3.3.2.40.1
Serbian (Cyrillic)	sr	2.16.840.1.113730.3.3.2.45.1
Serbian (Latin)	sh	2.16.840.1.113730.3.3.2.41.1
Slovakian	sk	2.16.840.1.113730.3.3.2.42.1
Slovenian	sl	2.16.840.1.113730.3.3.2.43.1
Spanish	es or es-ES	2.16.840.1.113730.3.3.2.15.1
Swedish	sv	2.16.840.1.113730.3.3.2.46.1
Turkish	tr	2.16.840.1.113730.3.3.2.47.1
Ukrainian	uk	2.16.840.1.113730.3.3.2.48.1

D.3. SUPPORTED LANGUAGE SUBTYPES

Language subtypes can be used by clients to determine specific values for which to search. For more information on using language subtypes, see Section 3.1.3.5, “Adding an Attribute Subtype” . Table D.2, “Supported Language Subtypes” lists the supported language subtypes for Directory Server.

Table D.2. Supported Language Subtypes

Language Tag	Language
af	Afrikaans
be	Belorussian
bg	Bulgarian
ca	Catalan
cs	Czechoslovakian

Language Tag	Language
da	Danish
de	German
el	Greek
en	English
es	Spanish
eu	Basque
fi	Finnish
fo	Faroese
fr	French
ga	Irish
gl	Galician
hr	Croatian
hu	Hungarian
id	Indonesian
is	Icelandic
it	Italian
ja	Japanese
ko	Korean
nl	Dutch
no	Norwegian
pl	Polish
pt	Portuguese
ro	Romanian
ru	Russian
sk	Slovakian
sl	Slovenian
sq	Albanian
sr	Serbian
sv	Swedish

Language Tag	Language
tr	Turkish
uk	Ukrainian
zh	Chinese

D.4. SEARCHING AN INTERNATIONALIZED DIRECTORY

When performing search operations, the Directory Server can sort the results based on any language for which the server has a supporting collation order. For a listing of the collation orders supported by the directory, see Section D.2, "Supported Locales".

NOTE

An LDAPv3 search is required to perform internationalized searches. Therefore, do not set the LDAPv2 option on the call for **ldapsearch**.

This section focuses using matching rule filters to return international attribute values. For more information on general **ldapsearch** syntax, see Section 10.4, "LDAP Search Filters". For information on searching internationalized directories using the **Users and Groups** portion of the Red Hat Console, see the online help.

- Section D.4.1, "Matching Rule Formats"
- Section D.4.2, "Supported Search Types"
- Section D.4.3, "International Search Examples"

D.4.1. Matching Rule Formats

The matching rule filters for internationalized searches can be represented in any several ways, and which one should be used is a matter of preference:

- As the OID of the collation order for the locale on which to base the search.
- As the language tag associated with the collation order on which to base the search.
- As the OID of the collation order and a suffix that represents a relational operator.
- As the language tag associated with the collation order and a suffix that represents a relational operator.

The syntax for each of these options is discussed in the following sections:

- Section D.4.1.1, "Using an OID for the Matching Rule"
- Section D.4.1.2, "Using a Language Tag for the Matching Rule"
- Section D.4.1.3, "Using an OID and Suffix for the Matching Rule"
- Section D.4.1.4, "Using a Language Tag and Suffix for the Matching Rule"

D.4.1.1. Using an OID for the Matching Rule

Each locale supported by the Directory Server has an associated collation order OID. For a list of locales supported by the directory server and their associated OIDs, see Table D.1, "Supported Locales".

The collation order OID can be used in the matching rule portion of the matching rule filter as follows:

```
attr:OID:=(relational_operator value)
```

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search for all ***departmentNumber*** attributes that are at or after **N4709** in the Swedish collation order, use the following filter:

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

D.4.1.2. Using a Language Tag for the Matching Rule

Each locale supported by the Directory Server has an associated language tag. For a list of locales supported by the directory server and their associated language tags, see Table D.1, "Supported Locales".

The language tag can be used in the matching rule portion of the matching rule filter as follows:

```
attr:language-tag:=(relational_operator value)
```

The relational operator is included in the value portion of the string, separated from the value by a single space. For example, to search the directory for all description attributes with a value of **estudiante** using the Spanish collation order, use the following filter:

```
cn:es:== estudiante
```

D.4.1.3. Using an OID and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, append a suffix that represents a specific operator to the OID in the matching rule portion of the filter. Combine the OID and suffix as follows:

```
attr: OID+suffix:=value
```

For example, to search for ***businessCategory*** attributes with the value **softwareprodukte** in the German collation order, use the following filter:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
```

The **.3** in the previous example is the equality suffix.

For a list of locales supported by the Directory Server and their associated OIDs, see Table D.1, "Supported Locales". For a list of relational operators and their equivalent suffixes, see Table D.3, "Search Types, Operators, and Suffixes".

D.4.1.4. Using a Language Tag and Suffix for the Matching Rule

As an alternative to using a relational operator-value pair, append a suffix that represents a specific operator to the language tag in the matching rule portion of the filter. Combine the language tag and suffix as follows:

```
attr: language-tag+suffix:=value
```

For example, to search for all surnames that come at or after **La Salle** in the French collation order, use the following filter:

```
sn:fr.4:=La Salle
```

For a list of locales supported by the Directory Server and their associated language tags, see Table D.1, "Supported Locales". For a list of relational operators and their equivalent suffixes, see Table D.3, "Search Types, Operators, and Suffixes".

D.4.2. Supported Search Types

The Directory Server supports the following types of international searches:

- equality (=)
- substring (*)
- greater-than (>)
- greater-than or equal-to (>=)
- less-than (<)
- less-than or equal-to (<=)

Approximate, or phonetic, and presence searches are supported only in English.

As with a regular **ldapsearch** search operation, an international search uses operators to define the type of search. However, when invoking an international search, either use the standard operators (=, >=, >, <, <=) in the value portion of the search string, or use a special type of operator, called a suffix (not to be confused with the directory suffix), in

the matching rule portion of the filter. Table D.3, "Search Types, Operators, and Suffixes" summarizes each type of search, the operator, and the equivalent suffix.

Table D.3. Search Types, Operators, and Suffixes

Search Type	Operator	Suffix
Less-than	<	.1
Less-than or equal-to	<=	.2
Equality	=	.3
Greater-than or equal-to	>=	.4
Greater-than	>	.5
Substring	*	.6

D.4.3. International Search Examples

The following sections show examples of how to perform international searches on directory data. Each example gives all the possible matching rule filter formats so that you can become familiar with the formats and select the one that works best.

D.4.3.1. Less-Than Example

Performing a locale-specific search using the less-than operator (<), or suffix (**.1**) searches for all attribute values that come before the given attribute in a specific collation order.

For example, to search for all surnames that come before the surname **Marquez** in the Spanish collation order, any of the following matching rule filters would work:

```
sn:2.16.840.1.113730.3.3.2.15.1:=< Marquez
...
sn:es:=< Marquez
...
sn:2.16.840.1.113730.3.3.2.15.1.1:=Marquez
...
sn:es.1:=Marquez
```

D.4.3.2. Less-Than or Equal-to Example

Performing a locale-specific search using the less-than or equal-to operator (<=), or suffix (**.2**) searches for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all room numbers that come at or before room number **CZ422** in the Hungarian collation order, any of the following matching rule filters would work:

```
roomNumber:2.16.840.1.113730.3.3.2.23.1:=<= CZ422
...
roomNumber:hu:=<= CZ422
...
roomNumber:2.16.840.1.113730.3.3.2.23.1.2:=CZ422
...
roomNumber:hu.2:=CZ422
```

D.4.3.3. Equality Example

Performing a locale-specific search using the equal to operator (=), or suffix (**.3**) searches for all attribute values that match the given attribute in a specific collation order.

For example, to search for all ***businessCategory*** attributes with the value **softwareprodukte** in the German collation order, any of the following matching rule filters would work:

```
businessCategory:2.16.840.1.113730.3.3.2.7.1:==softwareprodukte
```

```
...
businessCategory:de:== softwareprodukte
...
businessCategory:2.16.840.1.113730.3.3.2.7.1.3:=softwareprodukte
...
businessCategory:de.3:=softwareprodukte
```

D.4.3.4. Greater-Than or Equal-to Example

Performing a locale-specific search using the greater-than or equal-to operator (**>=**), or suffix (**.4**) searches for all attribute values that come at or after the given attribute in a specific collation order.

For example, to search for all localities that come at or after **Québec** in the French collation order, any of the following matching rule filters would work:

```
locality:2.16.840.1.113730.3.3.2.18.1:=>= Québec
...
locality:fr:=>= Québec
...
locality:2.16.840.1.113730.3.3.2.18.1.4:=Québec
...
locality:fr.4:=Québec
```

D.4.3.5. Greater-Than Example

Performing a locale-specific search using the greater-than operator (>), or suffix (**.5**) searches for all attribute values that come at or before the given attribute in a specific collation order.

For example, to search for all mail hosts that come after host **schranka4** in the Czechoslovakian collation order, any of the following matching rule filters would work:

```
mailHost:2.16.840.1.113730.3.3.2.5.1:=> schranka4
...
mailHost:cs:=> schranka4
...
mailHost:2.16.840.1.113730.3.3.2.5.1.5:=schranka4
...
mailHost:cs.5:=schranka4
```

D.4.3.6. Substring Example

Performing an international substring search searches for all values that match the given pattern in the specified collation order.

For example, to search for all user IDs that end in **ming** in the Chinese collation order, any of the following matching rule filters would work:

```
uid:2.16.840.1.113730.3.3.2.49.1:=* *ming
...
uid:zh:=* *ming
...
uid:2.16.840.1.113730.3.3.2.49.1.6:=* *ming
..
uid:zh.6:=* *ming
```

Substring search filters that use DN-valued attributes, such as ***modifiersName*** or ***memberOf***, do not always match entries correctly if the filter contains one or more space characters.

To work around this problem, use the entire DN in the filter instead of a substring, or ensure that the DN substring in the filter begins at an RDN boundary; that is, make sure it starts with the *type***=** part of the DN. For example, this filter should not be used:

```
(memberOf=*Domain Administrators*)
```

But either one of these will work correctly:

```
(memberOf=cn=Domain Administrators*)
...
(memberOf=cn=Domain Administrators,ou=Groups,dc=example,dc=com)
```

D.5. TROUBLESHOOTING MATCHING RULES

International collation order matching rules may not behave consistently. Some forms of matching-rule invocation do not work correctly, producing incorrect search results. For example, the following rules do not work:

```
ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w secret -b "dc=example,dc=com"
"sn:2.16.840.1.113730.3.3.2.7.1:=passin"

ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w secret -b "dc=example,dc=com"
"sn:de:=passin"
```

However, the rules listed below will work (note the **.3** before the **passin** value):

```
ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w secret -b "dc=example,dc=com"
"sn:2.16.840.1.113730.3.3.2.7.1.3:=passin"

ldapsearch -x -p 389 -D "uid=userID,ou=people,dc=example,dc=com" -w secret -b "dc=example,dc=com"
"sn:de.3:=passin"
```

APPENDIX E. MANAGING THE ADMIN SERVER

E.1. INTRODUCTION TO RED HAT ADMIN SERVER

Identity management and directory services with Red Hat Directory Server use three components, working in tandem:

- A Java-based management console
- An administration server which also functions as a web server
- An LDAP directory server

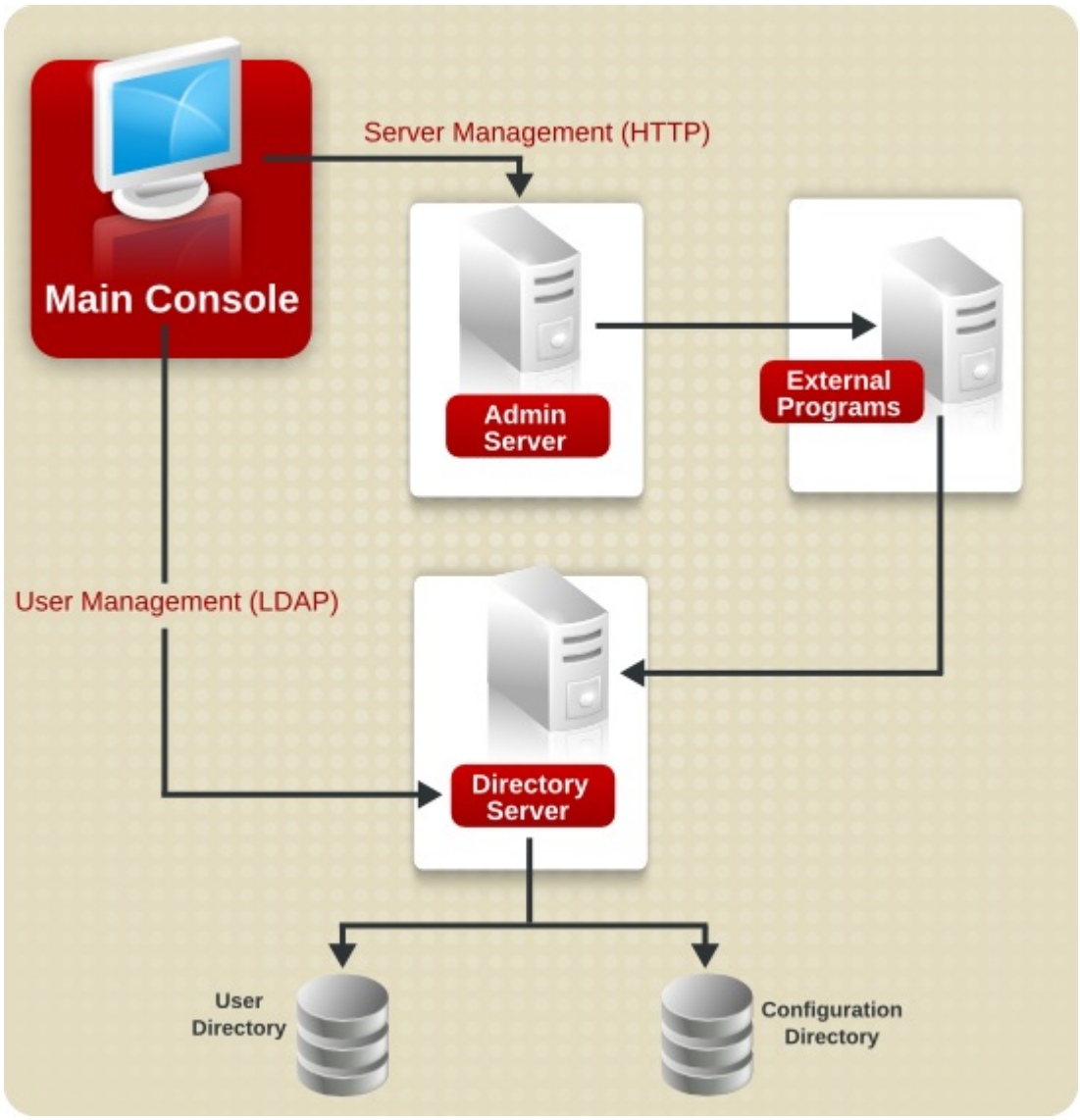


Figure E.1. Interactions between the Console, Admin Server and Directory Server

The Admin Server processes configuration requests for Directory Server instances and performs many common server tasks, such as stopping and starting server instances. Directory services are usually divided into two categories: *configuration* databases which store the Console and Admin Server settings and some Directory Server configuration and *user* databases which contain user and group information. These databases can be kept in the same Directory Server instance, but it is also possible to break these services into separate Directory Server instances. In that case, a Directory Server instance's configuration are stored in a separate Directory Server, called the *Configuration Directory Server*, and user data is stored in the *User Directory Server*. Because the Admin Server processes server configuration requests for Red Hat Directory Server, the Configuration Directory Server and User Directory Server instances are both defined in the Admin Server configuration.

As a web server, the Admin Server provides all of the online functions of the Directory Server, including handling connections to the Console and hosting web applications such as Admin Express. Clients connect to the Admin Server both over secure and standard connections, since the Admin Server supports both HTTP or HTTPS, if SSL/TLS is enabled.

When Red Hat Directory Server or Red Hat Certificate System (which depends on Red Hat Directory Server) is installed, then the Admin Server is automatically installed and configured as well. There can be multiple Directory Server instances and multiple Certificate System subsystems on a single machine, and all use the same instance of Admin Server.

There can be *only one* Admin Server per machine. This single Admin Server instance can handle multiple instances of Directory Server and other clients which can use the Admin Server, like Red Hat Certificate System.

When the Console is opened to manage an instance of Directory Server or Certificate System, even if the Console is on a different machine than the server instance being managed, it contacts the local Admin Server instance to perform the requested tasks. For example, Admin Server can execute programs to modify the server and application settings that are stored in the configuration directory or to change the port number that a server listens to.

The Admin Server itself can be managed through its own Java-based interface, by editing its configuration files, or through command-line tools.

E.2. ADMIN SERVER CONFIGURATION

The Admin Server is a separate server from Red Hat Directory Server or Red Hat Certificate System, although they work interdependently. The Admin Server processes, file locations, and configuration options are also separate. This chapter covers the Admin Server information, including starting and stopping the Admin Server, enabling SSL, viewing logs, and changing Admin Server configuration properties, such as the server port number.

E.2.1. Directory Server File Locations

Red Hat Admin Server conforms to the Filesystem Hierarchy Standards. For more information on FHS, see the FHS homepage, http://www.pathname.com/fhs/.

There are slight difference in the file locations depending on the platform, so the default Red Hat Enterprise Linux FHS locations (used in the examples) may not match every installation. Some platforms treat the Admin Server as optional software and therefore, under FHS, store Admin Server files in **/opt** directories.

The files and directories installed with Directory Server are listed in the tables below for each supported platform.

Table E.1. Red Hat Enterprise Linux 4 and 5 (x86 and x86_64)

File or Directory	Location
Log files	**/var/log/dirsrv/admin-serv/**
Configuration files	**/etc/dirsrv/admin-serv/**
Instance directory	**/usr/lib/dirsrv/admin-serv/**
Database files	**/var/lib/dirsrv/admin-serv/**
Runtime files	**/var/lock/dirsrv/admin-serv.*** **/var/run/dirsrv/admin-serv.***
Init scripts	**/etc/rc.d/init.d/dirsrv-admin/** **/etc/sysconfig/dirsrv-admin**
Tools	**/usr/bin/** **/usr/sbin/**

E.2.2. Starting and Stopping the Admin Server

The Admin Server is running when the **setup-ds-admin.pl** configuration script completes. Avoid stopping and starting the server to prevent interrupting server operations.

- When starting in SSL, the start script prompts for the password for the security (SSL certificate) database. It is possible to restart in SSL without being prompted for a password by using a password file. See Section E.2.9.4, “Creating a Password File for the Admin Server” for more information.

If there is not password file, then the Admin Server cannot be restarted in SSL through the Console, only the command-line scripts.

- Rebooting the host system can automatically start the Admin Server's **httpd** process. The directory provides startup or run command (**rc**) scripts. On Red Hat Enterprise Linux, use the **chkconfig** command to enable the Admin Server to start on boot.

E.2.2.1. Starting and Stopping Admin Server from the Console

1. Start the Console, and open the Admin Console.

```
/usr/bin/redhat-idm-console -a http://localhost:9830
```

2. In the **Tasks** tab, click **Restart Server** or **Stop Server**.

When the Admin Server is successfully started or stopped from the Console, the server displays a message box stating that the server has either started or shut down.

E.2.2.2. Starting and Stopping Admin Server from the Command Line

There are two ways to start, stop, or restart the Admin Server:

- There are scripts in the **/usr/sbin** directory.

```
/usr/sbin/{start|stop|restart}-ds-admin
```

- The Admin Server service can also be stopped and started using system tools on Red Hat Enterprise Linux 6 (64-bit) using the **service** command. For example:

```
service dirsrv-admin {start|stop|restart}
```

NOTE

The service name for the Admin Server process on Red Hat Enterprise Linux 6 (64-bit) is **dirsrv-admin**.

E.2.3. Opening the Admin Server Console

There is a simple script to launch the main Console. On Red Hat Enterprise Linux, run the following:

```
/usr/bin/redhat-idm-console
```

When the login screen opens, the Admin Server prompts for the user name, password, and Admin Server location. The Admin Server location is a URL; for a standard connection, this has the **http:** prefix for a standard HTTP protocol. If SSL/TLS is enabled, then this uses the **https:** prefix for the secure HTTPS protocol.

Figure E.2. Login Box

NOTE

It is possible to send the Admin Server URL and port with the start script. For example:

```
/usr/bin/redhat-idm-console -a http://localhost:9830
```

The **a** option is a convenience, particularly for logging into a Directory Server for the first time. On subsequent logins, the URL is saved. If the Admin Server port number is not passed with the **redhat-idm-console** command, then the server prompts for it at the Console login screen.

This opens the main Console window. To open the Admin Server Console, select the Admin Server instance from the server group on the left, and then click the **Open** at the top right of the window.

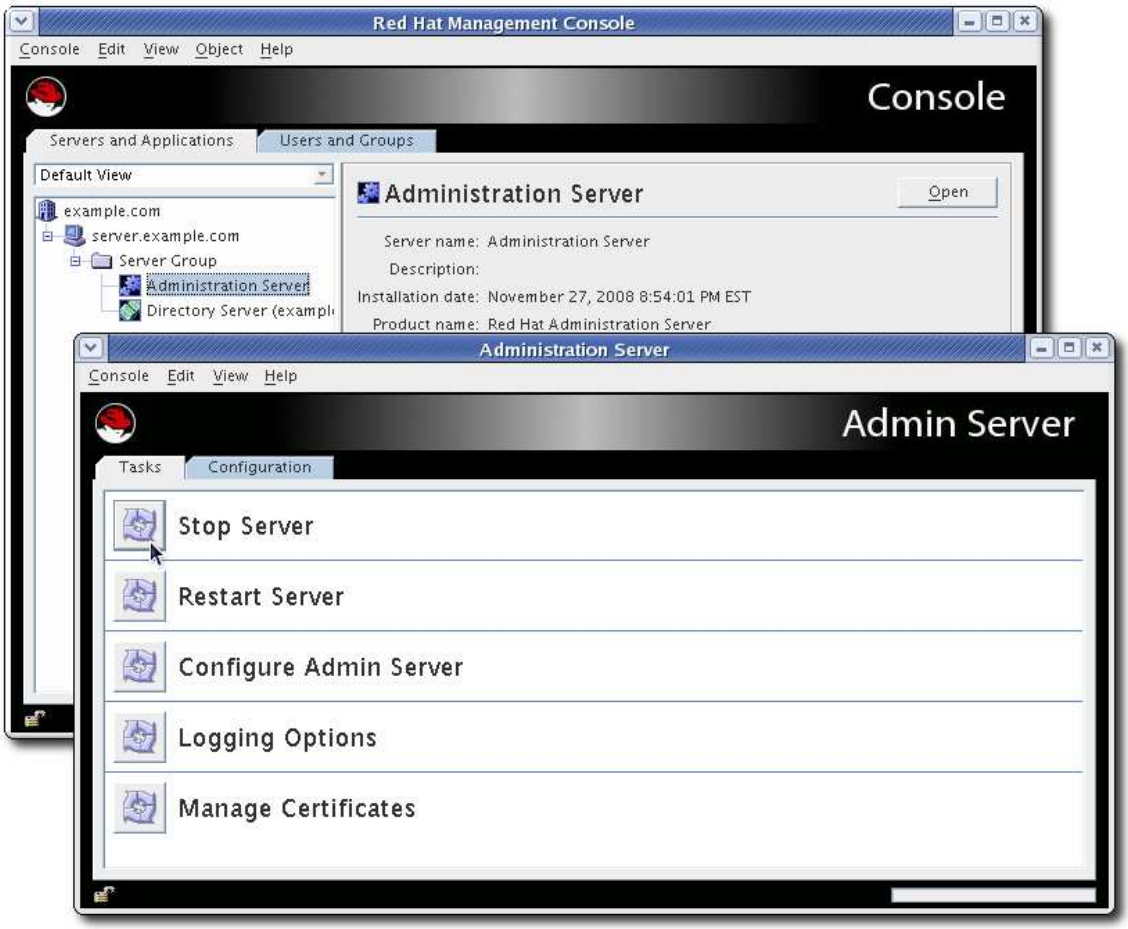


Figure E.3. The Admin Server Console

NOTE

Make sure that the Oracle Java Runtime Environment (JRE) or OpenJDK version 1.8.0 is set in the **PATH** before launching the Console. Run the following to see if the Java program is in the **PATH** and to get the version and vendor information:

```
java -version
```

E.2.4. Viewing Logs

Log files monitor activity for Admin Server and can help troubleshoot server problems. Admin Server logs use the Common Logfile Format, a broadly supported format that provides information about the server.

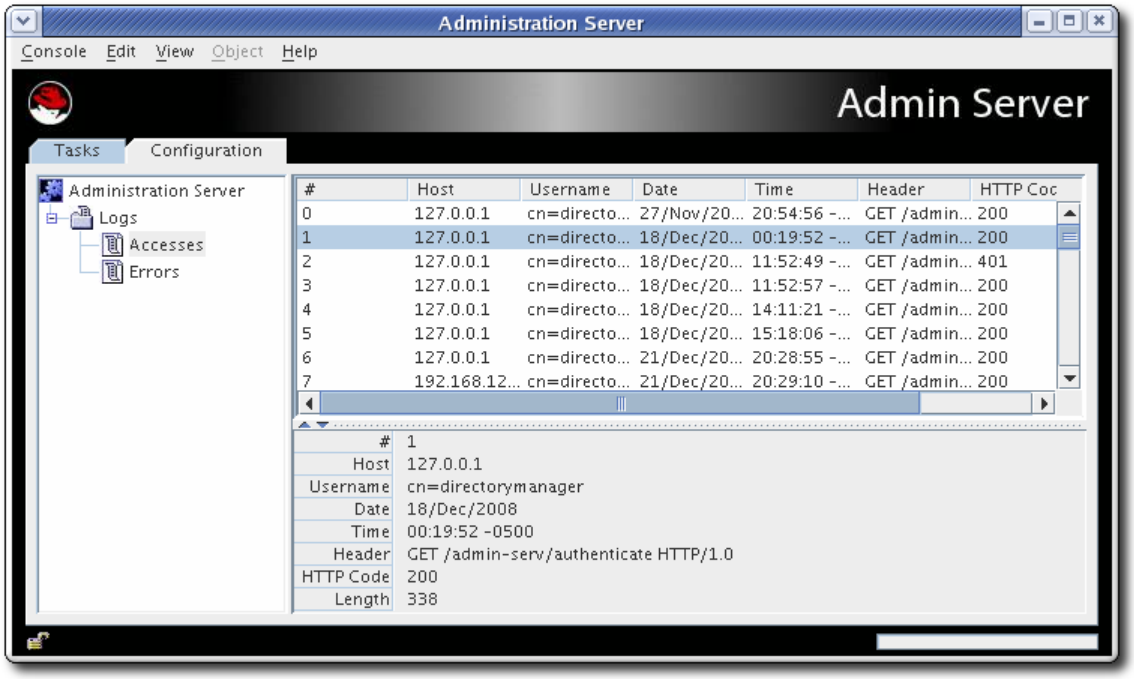
Admin Server generates two kinds of logs:

- *Access logs*. Access logs show requests to and responses from the Admin Server. By default, the file is located at **/var/log/dirsrv/admin-servaccess**.
- *Error logs*. Error logs show messages for errors which the server has encountered since the log file was created. It also contains informational messages about the server, such as when the server was started and who tried unsuccessfully to log on to the server. By default, the file is located at **/var/log/dirsrv/admin-serverror**.

The logs can be viewed through Admin Server Console or by opening the log file.

E.2.4.1. Viewing the Logs through the Console

1. Open the Admin Server management window.
2. Click the **Configuration** tab.
3. Expand the **Logs** directory, and click the log file name, either **Accesses** or **Error**.



E.2.4.2. Viewing Logs in the Command Line

The access log, by default, is at **/var/log/dirsrv/admin-servaccess**. To view the access log, open it in an editor such as **vi**.

Access logs show connections to the Admin Server based on the IP address of the client, the user name, and the method that the request was sent. Each line has the following format:

> *ip_address - bind_DN* [*timestamp* -0500] "GET|POST *cgi*" *HTTP_response bytes*

Example logs are shown in Example E.1, “Example Access Logs” .

> **Example E.1. Example Access Logs**
>
> 127.0.0.1 - cn=directory manager [23/Dec/2008:19:32:52 -0500] "GET /admin-serv/authenticate HTTP/1.0" 200 338
> 192.168.123.121 - cn=directory manager [23/Dec/2008:19:33:14 -0500] "POST /admin-serv/tasks/Configuration/ServerSetup HTTP/1.0" 200 244
> 192.168.123.121 - cn=directory manager [23/Dec/2008:19:33:16 -0500] "GET /admin-serv/tasks/Configuration/ReadLog?op=count&name=access HTTP/1.0" 200 10

The error log, by default, is at **/var/log/dirsrv/admin-serverrors**. To view the error log, open it in an editor such as **vi**.

Error logs record any problem response from the Admin Server. Like the access log, error logs also records entries based the client's IP address, along with the type of error message, and the message text:

> [*timestamp*] [*severity*] [client *ip_address error_message*

The *severity* message indicates whether the error is critical enough for administrator intervention. **[warning]**, **[error]**, and **[critical]** require immediate administrator action. Any other severity means the error is informational or for debugging.

Example logs are shown in Example E.2, “Example Error Logs” .

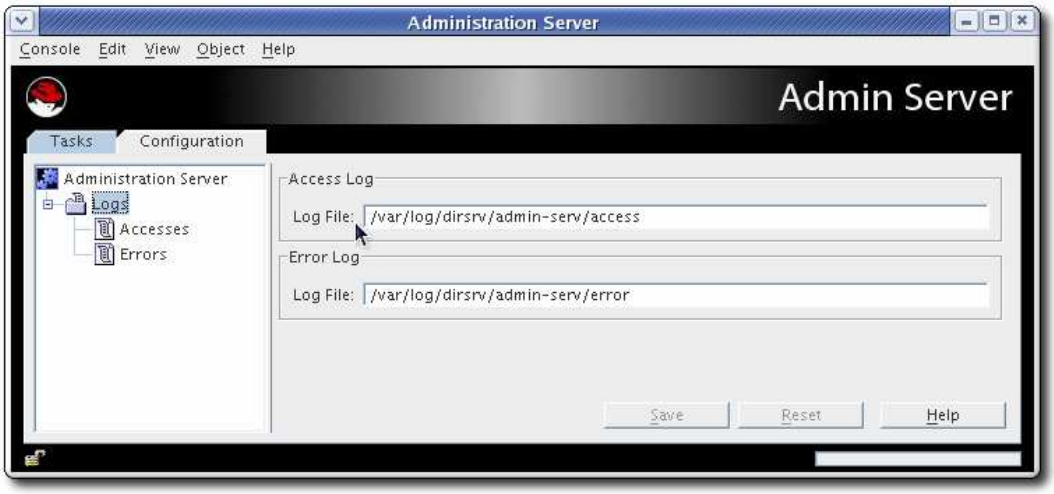
> **Example E.2. Example Error Logs**
>
> [Mon Dec 22 23:44:59 2008] [notice] [client 127.0.0.1] admserv_host_ip_check: ap_get_remote_host could not resolve 127.0.0.1
> [Mon Dec 22 23:44:59 2008] [notice] [client 127.0.0.1] admserv_host_ip_check: host [localhost.localdomain] did not match pattern [*.example.com] -will scan aliases
> [Mon Dec 22 23:44:59 2008] [notice] [client 127.0.0.1] admserv_host_ip_check: host alias [localhost] did not match pattern [*.example.com]
> [Mon Dec 22 23:44:59 2008] [notice] [client 127.0.0.1] admserv_check_authz(): passing [/admin-

```
serv/authenticate] to the userauth handler
[Mon Dec 22 23:45:16 2008] [notice] [client 192.168.123.121] admserv_host_ip_check: ap_get_remote_host
could not resolve 192.168.123.121
```

E.2.4.3. Changing the Log Name in the Console

The access and error log files' names can be changed to rotate the files. This rotation has to be done manually to create new files if the existing log files become too large.

1. Open the Admin Server management window.
2. Click the **Configuration** tab.
3. Click **Logs** in the left panel.
4. In the **Logs** window on the right, enter the new log file name.



> **WARNING**
>
> The path to the log file is absolute and cannot be changed.

5. Click **OK** to save the changes.
6. Open the **Tasks** tab, and click the **Restart Server** button to restart the server and apply the changes.

E.2.4.4. Changing the Log Location in the Command Line

The access and error log files' names and locations can be changed to rotate the files. This rotation has to be done manually to create new files if the existing log files become too large. The location can be changed if the default location in **/var/log/dirsrv/admin-serv** does not meet the application needs.

The Admin Server configuration is stored in two locations. The main entry is an LDAP entry in the Configuration Directory Server's **o=NetscapeRoot** database. The other is the **console.conf** file. Changing the log settings requires changing both settings.

1. Edit the Admin Server configuration entry in the Configuration Directory Server.
    1. Get the name of the Admin Server entry. Since the Admin Server entry has a special object class, **nsAdminConfig**, it is possible to search for the entry using that object class to retrieve the DN.

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b
"o=NetscapeRoot" "(objectclass=nsAdminConfig)" dn
```

```
version:1
dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
```

2. The Admin Server entry can be edited using **ldapmodify**. The access and error log settings are stored in the ***nsAccessLogs*** and ***nsErrorLogs*** attributes, respectively. For example:

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
changetype:modify
replace:nsAccessLog
nsAccessLog:/var/log/dirsrv/admin-serv/access_new
```

Hit **Enter** twice to submit the operation, and then **Control+C** to close **ldapmodify**.

2. Open the Admin Server configuration directory.

```
cd /etc/dirsrv/admin-serv
```

3. Edit the **console.conf** file. For the access log, edit the path and filename in the **CustomLog** parameter. For the error log, edit the path and filename in the **ErrorLog** parameter.

```
CustomLog /var/log/dirsrv/admin-serv/access_new common
ErrorLog /var/log/dirsrv/admin-serv/error_new
```

Leave the term **common** after the access log path; this means that the access log is in the Common Log Format.

4. Restart the Admin Server.

```
service dirsrv-admin restart
```

E.2.4.5. Setting the Logs to Show Hostnames Instead of IP Addresses

By default, the logs show the IP address of the clients which connect to the Admin Server. This is faster for the Admin Server, since it does not have to do a DNS lookup for every connection. It is possible to set the Admin Server to perform a DNS lookup so that host names are used in the logs. Along with being friendlier to read and search, using host names instead of IP addresses also removes some unnecessary error messages about being unable to resolve host names.

To configure the Admin Server to perform DNS lookups:

1. Edit the **console.conf** file for the Admin Server.

```
cd /etc/dirsrv/admin-serv
vim console.conf
```

2. Set the **HostnameLookups** parameter to **on**. By default, this is turned off, so that IP addresses are recorded in logs instead of host names.

```
HostnameLookups on
```

E.2.5. Changing the Port Number

The *port number* specifies where an instance of Admin Server listens for messages.

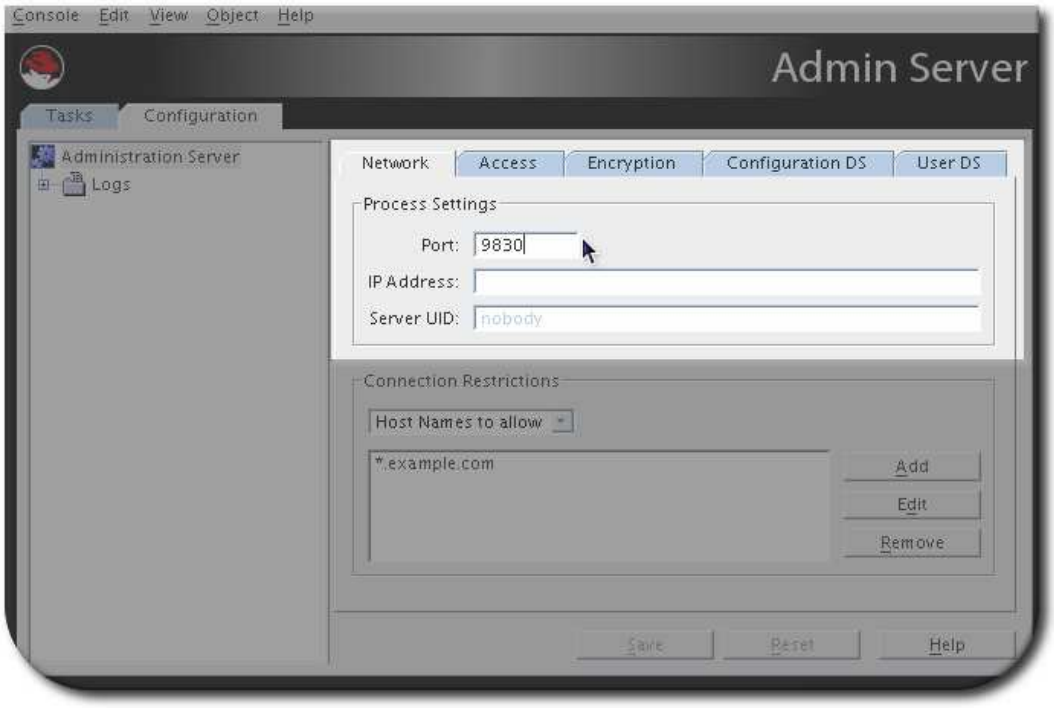
The default port number for Admin Server is set when the instance is first installed and the configuration script, such as **setup-ds-admin.pl**, is run. The default port number is **9830**, although if that number is in use, then the setup program will use a randomly-generated number larger than **1024** or one can assign any port number between **1025** and **65535**.

E.2.5.1. Changing the Port Number in the Console

1. Open the Admin Server management window.

2. Click the **Configuration** tab.

3. Click the **Network** tab.



4. Enter the port number for the Admin Server instance in the **Port** field. The Admin Server port number has a default number of **9830**.

5. Click **OK**.

6. Open the **Tasks** tab, and click the **Restart Server** button to restart the server and apply the changes.

7. Close the Console, and then restart the Console, specifying the new Admin Server port number in the connection URL.

E.2.5.2. Changing the Port Number in the Command Line

The port number for the Admin Server is **9830** by default.

The Admin Server configuration is stored in two locations. The main entry is an LDAP entry in the Configuration Directory Server's **o=NetscapeRoot** database. The other is the **console.conf** file. Changing the port number requires changing both settings.

1. Edit the Admin Server configuration entry in the Configuration Directory Server.

    1. Get the name of the Admin Server entry. Since the Admin Server entry has a special object class, **nsAdminConfig**, it is possible to search for the entry using that object class to retrieve the DN.

    ```
    ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b
    "o=NetscapeRoot" "(objectclass=nsAdminConfig)" dn

    version:1
    dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
    Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
    ```

    2. The Admin Server entry can be edited using **ldapmodify**. The port number is set in the ***nsServerPort*** attribute. For example:

    ```
    ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

    dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
    Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
    changetype:modify
    replace:nsServerPort
    nsServerPort:10030
    ```

Hit **Enter** twice to submit the operation, and then **Control+C** to close **ldapmodify**.

2. Open the Admin Server configuration directory.

```
cd /etc/dirsrv/admin-serv
```

3. Edit the **Listen** parameter in the **console.conf** file.

```
Listen 0.0.0.0:10030
```

4. Restart the Admin Server.

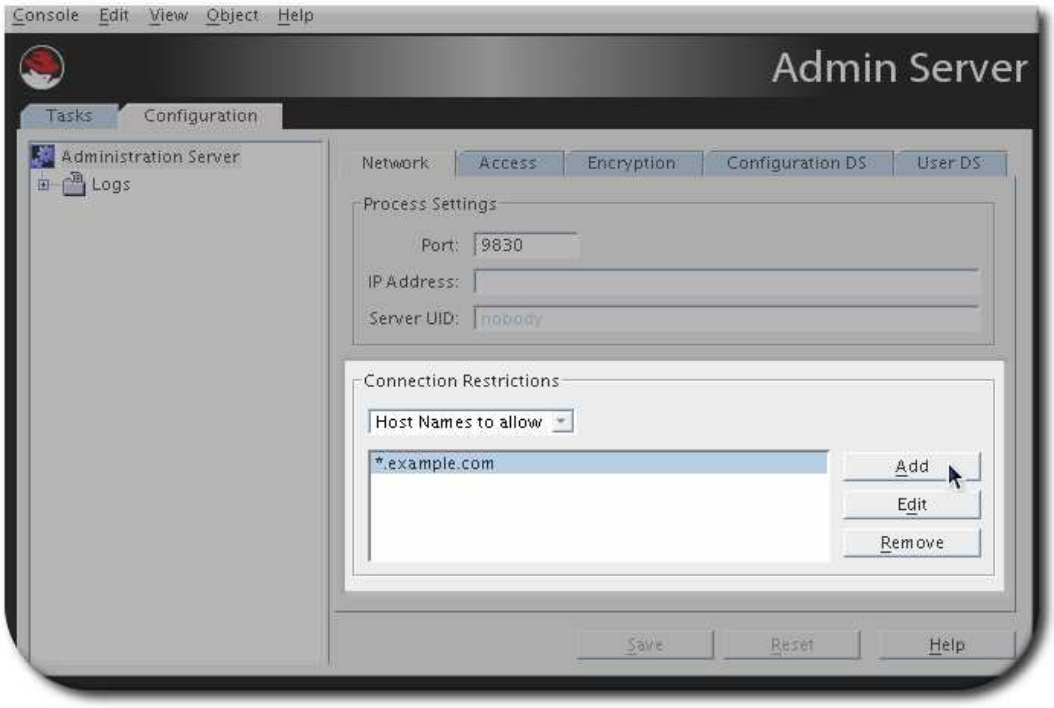
```
service dirsrv-admin restart
```

E.2.6. Setting Host Restrictions

Connection restrictions specify which hosts are allowed to connect to the Admin Server. You can list these hosts by DNS name, IP address, or both. Only host machines listed within the connection restriction parameters are allowed to connect to the Admin Server. This setting allows wildcards within a domain or an IP address range to make setting connection restrictions simpler.

E.2.6.1. Setting Host Restrictions in the Console

1. Open the Admin Server management window.
2. Click the **Configuration** tab.
3. Click the **Network** tab.
4. The **Connection Restrictions** area displays a list of hosts allowed to connect to the Admin Server. The drop-down list specifies whether the list entries are added by DNS name or by IP address. The list is evaluated first by host names, and then by IP addresses.



5. Click the **Add** button to add another host to the list of allowed computers. To add a host name, make sure the drop-down list at the top reads **Host Names to allow**; to add an IP address, select **IP Addresses to allow**.
6. Fill in the host information, either the host ame or an IPv4 or IPv6 address.

The * wildcard can be used to specify a group of hosts. For instance, ***.example.com** allows all machines in the **example.com** domain to access the instance. Entering **205.12.*.** allows all hosts whose IP addresses begin with **205.12** to access the instance.

When specifying IP address restrictions, include all three separating dots. If you do not, the Admin Server returns an error message.

7. Click **OK** to close the **Add...** dialog box, and then click the **Save** button to save the new host.
8. Open the **Tasks** tab, and click the **Restart Server** button to restart the server and apply the changes.

To change the information for a host or IP address listed, click the **Edit** button and change the given information. To remove an allowed host or IP address, select the host from the list, and click **Remove**. **Admin Server**.

E.2.6.2. Setting Host Restrictions in the Command Line

Host restrictions sets rules for what network clients can connect to the Admin Server and, therefore, to services which use the Admin Server. There are two kinds of host restrictions, restrictions based on the host or domain name and restrictions based on the IP address.

The Admin Server host restrictions are set in the main configuration entry in the Configuration Directory Server's **o=NetscapeRoot** database. There are two attributes for setting host restrictions, ***nsAdminAccessAddresses*** and ***nsAdminAccessHosts*** for IP addresses and host names, respectively.

NOTE

The Admin Server supports both IPv4 and IPv6 addresses.

The Admin Server entry can be edited using **ldapmodify**.

To set host restrictions:

1. Get the name of the Admin Server entry. Since the Admin Server entry has a special object class, **nsAdminConfig**, it is possible to search for the entry using that object class to retrieve the DN.

```
ldapsearch -D "cn=directory manager" -w secret -p 389 -h server.example.com -x -b "o=NetscapeRoot" "
(objectclass=nsAdminConfig)" dn

version:1
dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
```

2. To set IP address-based restrictions, edit the ***nsAdminAccessAddresses*** attribute. Either IPv4 or IPv6 addresses can be used.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
changetype:modify
replace:nsAdminAccessAddresses
nsAdminAccessAddresses:72.5.*.*
```

Hit **Enter** twice to submit the operation, and then **Control+C** to close **ldapmodify**.

The ***nsAdminAccessAddresses*** value can use wildcards to allow ranges. Either IPv4 or IPv6 addresses can be used.

For example, to allow all IP addresses:

```
nsAdminAccessAddresses:*
```

To allow only a subset of addresses on a local network:

```
nsAdminAccessAddresses:192.168.123.*
```

3. To set host name or domain-based restrictions, edit the ***nsAdminAccessHosts*** attribute.

```
ldapmodify -D "cn=directory manager" -W -p 389 -h server.example.com -x

dn: cn=configuration,cn=admin-serv-example,cn=Red Hat Administration Server,cn=Server
Group,cn=server.example.com,ou=example.com,o=NetscapeRoot
changetype:modify
replace:nsAdminAccessHosts
nsAdminAccessHosts:*.example.com
```

Hit **Enter** twice to submit the operation, and then **Control+C** to close **ldapmodify**.

4. Restart the Admin Server to apply the changes.

```
service dirsrv-admin restart
```

E.2.7. Changing the Admin User's Name and Password

During installation, you are asked to enter a user name and password for the *Configuration Administrator*, the user authorized to access and modify the entire configuration directory. The Configuration Administrator entry is stored in the directory under the following DN:

```
uid=userID,ou=Administrators,ou=TopologyManagement,o=NetscapeRoot
```

The Configuration Administrator's user name and password are managed through the Directory Server and are represented in an LDAP entry; this is described in the *Directory Server Administrator's Guide*.

During installation, the Configuration Administrator's user name and password are used to automatically create the *Administration Server Administrator*. This user can perform a limited number of administrative tasks, such as starting, stopping, and restarting servers in a local server group. The Administration Server Administrator is created for the purpose of logging into the Console when the Directory Server is not running.

The Administration Server Administrator does not have an LDAP entry; it exists only as an entity in a local configuration file, **/etc/dirsrv/admin-serv/admpw**.

Even though they are created at the same time during installation, and are identical at that time, the Configuration Administrator and Administration Server Administrator are two separate entities. If you change the user name or password for one in the Console, the Console does not automatically make the same changes for the other.

The Administration Server Administrator has full access to all configuration settings in the Admin Server. The information for the admin user is set on the **Access** tab in the Console.

NOTE

The Admin Server administrator user name and password are stored in the **/etc/dirsrv/admin-serv/admpw** file. For example:

```
admin:{SHA}W6ph5Mm5Pz8GgiULbPgzG37mj9g=
```

The password is encrypted and cannot be changed directly in the **admpw** file. The user name can be changed in this file, but cannot be used to log into the Console unless the password is updated in the Console first. For this reason, it is better to edit the Administration Server Administrator user name and password only through the Admin Server Console.

To change the Administration Server Administrator's ID or password:

1. Open the Admin Server management window.
2. Click the **Configuration** tab.

3. Click the **Access** tab.

4. Change the admin user's name or password. The user name is the ID given for logging into the Admin Server.

5. Click **Save**.

E.2.8. Managing SELinux for the Admin Server

SELinux is a security function in Linux that categorizes files, directories, ports, processes, users, and other objects on the server. Each object is placed in an appropriate security context to define how the object is allowed to behave on the server through its role, user, and security level. These roles for objects are grouped in domains, and SELinux rules define how the objects in one domain are allowed to interact with objects in another domain.

SELinux itself is much more complex to manage and implement than what is described here. This section is concerned only with giving the SELinux details for the Administration Server. For more general information about SELinux, see the Red Hat Enterprise Linux 6 Security-Enhanced Linux User Guide .

NOTE

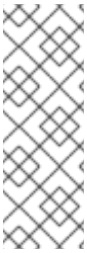
SELinux is a feature of Red Hat Enterprise Linux and, as such, is covered in the Red Hat Enterprise Linux *Scurity-Enhanced Guide* at https://access.redhat.com/site/documentation/en-US/Red_Hat_Enterprise_Linux/6/html/Security-Enhanced_Linux/index.html.

E.2.8.1. SELinux Definitions for the Admin Server

SELinux has three different levels of enforcement: disabled (no SELinux), permissive (where the rules are lax), and enforcing (where all rules are strictly enforced). Red Hat Directory Server and the Admin Server have defined SELinux policies that allow them to run as normal under strict SELinux enforcing mode.

By default, the Admin Server runs confined by SELinux policies.

There are two SELinux security contexts that apply to the Admin Server. When the server starts, it starts within its own Admin Server-specific SELinux domain, **dirsrvadmin_t**, where the Admin Server scripts are confined. However, the Admin Server process is simply the Apache web server daemon, **httpd**. So, once the Admin Server process is started, it transitions into the existing **httpd_t** domain on Red Hat Enterprise Linux.



NOTE

For the Admin Server to start with the process properly confined, it must be started or restarted using the **service** command. For example:

```
service dirsrv-admin restart
```

The **start-ds-admin** script is not supported by SELinux.

All CGIs invoked by the Admin Server, such as scripts for Admin Express, run in a special confined security domain, **httpd_dirsrvadmin_script_t**, which is separate from the **dirsrvadmin_t** or **httpd_t** domains.

Table E.2, "Summary of Admin Server SELinux Policies" lists the security contexts and domains for the major components of the Admin Server.

Table E.2. Summary of Admin Server SELinux Policies

File Path	Security Context	Description
dirsrvadmin_t Domain		
/usr/sbin/[start|stop|restart]-ds-admin	dirsrvadmin_exec_t	The Admin Server start, stop, and restart scripts
/etc/dirsrv/admin-serv/*	dirsrvadmin_config_t	Admin Server configuration files, such as **adm.conf**
httpd_dirsrvadmin_script_t Domain		
/usr/lib[64]/dirsrv/cgi-bin/*	httpd_dirsrvadmin_script_exec_t	The CGI scripts and files used by Admin Server web services, like Admin Express
httpd_t Domain[a]		
/var/log/dirsrv/admin-serv/*	httpd_log_t	The log files for the Admin Server
/var/run/dirsrv/admin-serv.*	httpd_var_run_t	The PID file for the Admin Server process
Ports 80, 443, and the Admin Server HTTP port (9830 by default)	http_port_t	The ports used by the Apache web server and the Admin Server web services, including the default HTTP and HTTPS Apache ports and whatever the configured HTTP port[b] for the Admin Server is

[a] There are more contexts configured by default within the httpd_t domain, but they are not relevant to the Admin Server SELinux policies.

[b] Only the HTTP port is configured for the Admin Server when it is set up, so only this port is added to the SELinux configuration automatically. The HTTPS port must be added manually, as described in Section 1.10.6, "Labeling SSL/TLS Ports".

The Admin Server SELinux policies are configured when the server is set up (when **setup-ds-admin.pl** or **register-ds-admin.pl** are run). These policies are removed when the Admin Server is uninstalled.

E.2.8.2. Viewing SELinux Policies for the Admin Server

All Admin Server policies are located in **/usr/share/selinux/strict/dirsrv-admin.pp**. The configured policies can be viewed using the SELinux Administration GUI.

1. Open the **Systems** menu.
2. Open the **Administration** menu, and select the **SELinux Management** item.

NOTE

You can launch the GUI from the command line using **system-config-selinux**.

To check the version of the Admin Server SELinux policy installed, click the **Policy Module** link.

To view the policies set on the individual files and processes, click the **File Labeling** link. To view the policies for the port assignments for the server, click the **Network Port** link.

E.2.8.3. Labeling SSL/TLS Ports

When the Admin Server is first set up, the given HTTP port is labeled for SELinux (the default is port 9830). However, SSL/TLS is set up separately, after the Admin Server is already configured, so the HTTPS port for the Admin Server is not automatically labeled.

The default HTTPS port, 443, is already labeled as part of the Apache web service policies. If the Admin Server uses a port other than the default for its SSL/TLS connections, however, then an administrator must label the port manually. This can be done in the SELinux administrative interface show in Section 1.10.3, "Viewing and Editing SELinux Policies for the Directory Server". It can also be done easily using the **semanage** script.

Use the **port** subcommand, the **-t** option to identify the security context, and the **-p** option to identify the port. The **-a** option adds the port label. For example:

```
semanage port -a -t http_port_t -p tcp 1443
```

To delete a port label, use the **-d** option. For example:

```
semanage port -d -t http_port_t -p tcp 1443
```

E.2.8.4. Starting the Admin Server Confined by SELinux

The Admin Server's **httpd** process initially starts in its own **dirsrvadmin_t**, and then transitions to the **http_t** domain after starting. This daemon only runs confined in the appropriate SELinux policies when the **service** command is used to run the Admin Server.

```
service dirsrv-admin start|stop|restart
```

E.2.9. Working with SSL

The Admin Server can run over HTTPS (secure HTTP) if SSL is enabled on the server. There are steps to enabling SSL:

1. Generating and submitting a certificate request.
2. Receiving and installing the certificate.
3. Trusting the certificate authority (CA) which issued the certificate.
4. Changing the Admin Server configuration to allow SSL connections.

E.2.9.1. Requesting and Installing a Server Certificate

The Admin Server Console has a tool, the **Certificate Request Wizard**, which generates a valid certificate request to submit to any certificate authority (CA).

1. In the Admin Server Console, select the **Tasks** tab, and click **Manage Certificates**.

2. Create a certificate request.
    1. Select the **Server Certs** tab, and click the **Request** button.

       Click **Next**.
    2. Enter the **Requester Information** in the blank text fields, then click **Next**.

- *Server Name.* The fully qualified host name of the Directory Server as it is used in DNS and reverse DNS lookups; for example, **server.example.com**. The server name is critical for client-side validation to work, which prevents man-in-the-middle attacks.

  IMPORTANT

  This *must* be a valid host name that can be resolved correctly by all Admin Server clients, or TLS/SSL will not work.

- *Organization.* The legal name of the company or institution. Most CAs require this information to be verified with legal documents such as a copy of a business license.
- *Organizational Unit. Optional.* A descriptive name for the organization within the company.
- *Locality. Optional.* The company's city name.
- *State or Province.* The full name of the company's state or province (no abbreviations).
- *Country.* The two-character abbreviation for the country's name (ISO format). The country code for the United States is US.

3. Enter the password that used to protect the private key, and click **Next**.

The **Next** button is grayed out until a password is supplied.

3. The **Request Submission** dialog box provides two ways to submit a request: directly to the CA (if there is one internally) or manually. To submit the request manually, select **Copy to Clipboard** or **Save to File** to save the certificate request which will be submitted to the CA.

To submit the request to a CA manually, either email it or use the web form for the CA, if one is available. Copy the certificate request information and submit it using the appropriate method.

```
-----BEGIN NEW CERTIFICATE REQUEST-----
MIIBrjCCARcCAQAwbjELMAkGA1UEBhMCVXMxEzARBgNVBAgTCkNBTElGT1J
OSUExLDAqBgVBAoTI25ldHNjYXBllGNvbW11bmljYXRpb25zIGNvcnBvcmF
0aW9uMRwwGgYDVQQDExNtZWxsb24ubmV0c2NhcGUuY29tMIGfMA0GCSqGSI
b3DQEBAQUAA4GNADCBiQKBgQCwAbskGh6SKYOgHy+UCSLnm3ok3X3u83Us7
ug0EfgSLR0f+K41eNqqRftGR83emqPLDOf0ZLTLjVGJaH4Jn4l1gG+JDf/n
/zMyahxtV7+mT8GOFFigFfuxaxMjr2j7IvELlxQ4lfZgWwqCm4qQecv3G+N
9YdbjveMVXW0v4XwIDAQABoAAwDQYK
------END NEW CERTIFICATE REQUEST-----
```

4. Wait for the CA to respond with the server certificate; this can be as short as a few hours for an internal CA or as long as several weeks for a third-party CA.

5. Save the issued certificate to a file.

NOTE

Keep a backup of the certificate data in a safe location. If the system ever loses the certificate data, the certificate can be reinstalled using the backup file.

6. Install the certificate.

    1. Select the **Tasks** tab, and click **Manage Certificates**.

    2. Select the **Server Certs** tab, and click **Install**.

    3. Give the absolute path to the certificate (*In this file* radio button) or paste the certificate text in the text box (*In the following encoded text block* radio button), then click **Next**.

    4. Check that the certificate information displayed is correct, and click **Next**.

    5. Name the certificate, and click **Next**.

    6. Provide the password that protects the private key. This password is the same as the one provided in step c.

After installing the server certificate, configure the Admin Server to trust the CA which issued the server's certificate.

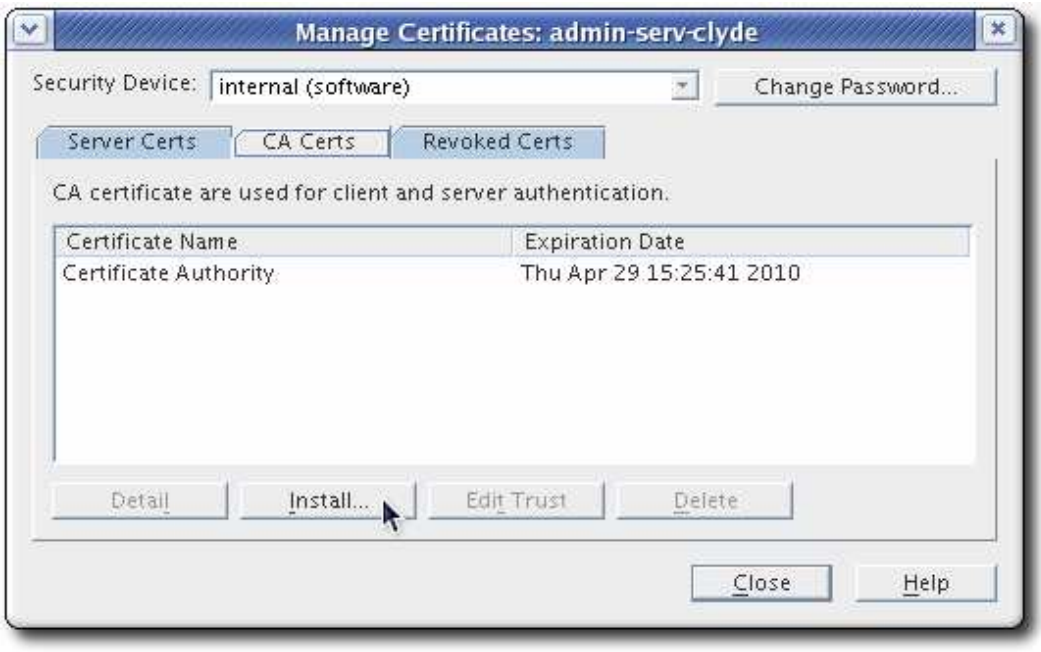
E.2.9.2. Installing a CA Certificate

To configure the Admin Server to trust the CA, obtain the CA's certificate and install it into the server's certificate database. Some commercial CAs provide a web site that allow users to automatically download the certificate, while others will email it back to users.

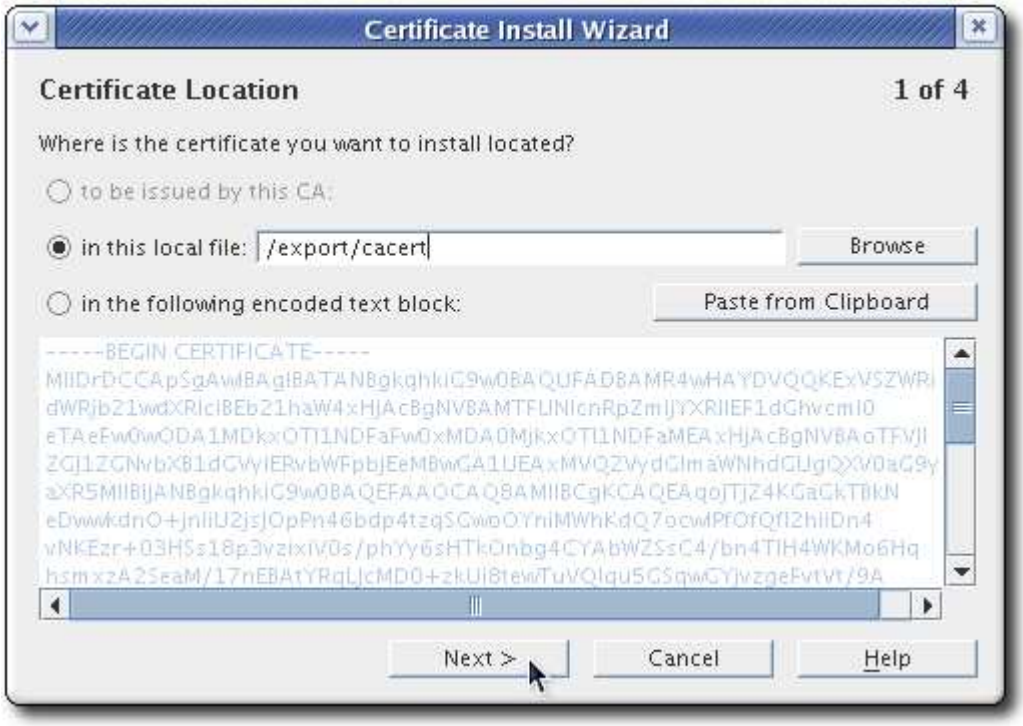
After receiving the CA certificate, use the **Certificate Install Wizard** to configure the Admin Server to trust the CA.

1. In the Admin Server Console, select the **Tasks** tab, and click **Manage Certificates**.

2. Go to the **CA Certs** tab, and click **Install**.



3. If the CA's certificate is saved to a file, enter the path in the field provided. Alternatively, copy and paste the certificate, including the headers, into the text box. Click **Next**.



4. Click **Next** to move through the panels that show the CA certificate information and the certificate name.
5. Select the purpose of trusting this certificate authority; it is possible to select both options:
    - *Accepting connections from clients (Client Authentication).* The server checks that the client's certificate has been issued by a trusted certificate authority.
    - *Accepting connections to other servers (Server Authentication).* This server checks that the directory to which it is making a connection (for replication updates, for example) has a certificate that has been issued by a trusted certificate authority.

6. Click **Done**.

After installing the CA certificate, it is listed in the **CA Certificates** tab in the Console.

NOTE

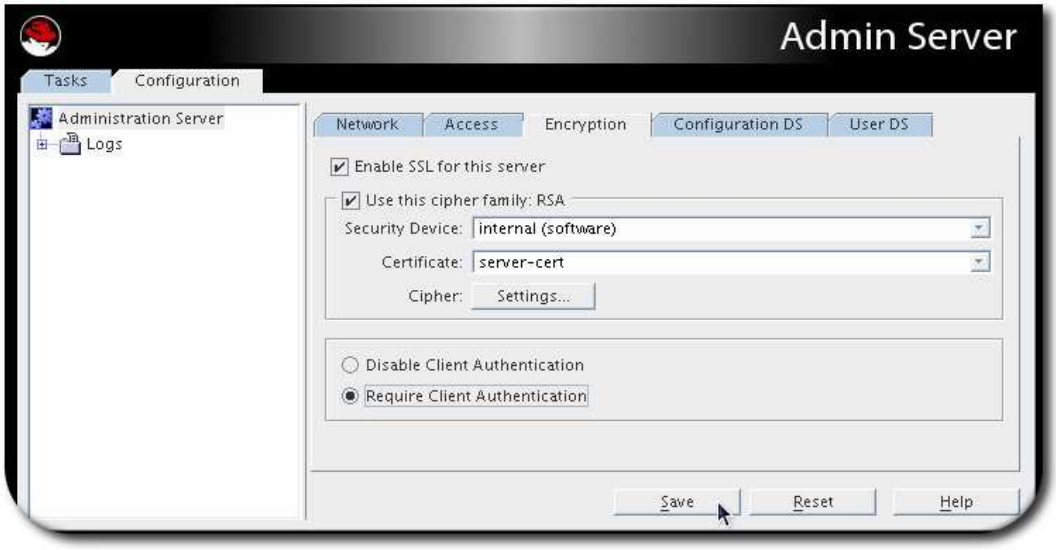
If a CA certificate is incorrectly generated, it is listed in the **Server Certificates** tab in the Console rather than the **CA Certificates** tab. The certificate still works as a CA certificate, even though it is listed in the wrong tab.

Still, request certificates from a real certificate authority to minimize the risk of using an incorrectly generated certificate and breaking SSL/TLS in the Admin Server.

E.2.9.3. Enabling SSL

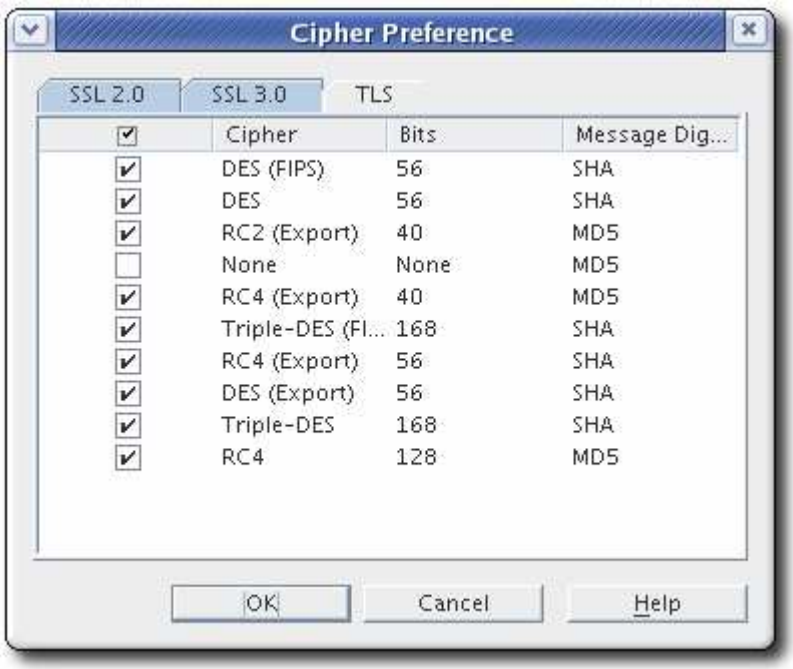
1. Open the Admin Server management window.

2. Click the **Configuration** tab.
3. Click the **Encryption** tab.



4. Select the **Enable SSL for this server** check box.
5. Select the **Use this cipher family: RSA** check box.
6. Choose the security device where the key is stored. By default, the key is stored in the local key database, **Internal (Software-based)**. If the key is stored on an external device (such as a smart card), select that device from the menu.
7. Choose the server certificate to use with SSL.

   The certificates available in the token certificate database are listed in the drop-down menu.
8. Click the **Settings** button to set the ciphers that the Admin Server accepts for SSL/TLS connections.



9. Set whether to require client authentication to the Admin Server. Client authentication means that the server checks that the client's certificate has been issued by a trusted CA.
10. Click **Save**.

E.2.9.4. Creating a Password File for the Admin Server

Normally, if SSL is enabled, the server prompts for a security password when the Admin Server is restarted:

```
Starting dirsrv-admin:
Please enter password for "internal" token:
```

The Admin Server can use a password file when TLS/SSL is enabled so that the server restarts silently, without prompting for the security password.

WARNING

This password is stored in clear text within the password file, so its usage represents a significant security risk. Do not use a password file if the server is running in an unsecured environment.

1. Open the Admin Server configuration directory.

   ```
   cd /etc/dirsrv/admin-serv
   ```

2. Create a password file named **password.conf**. The file should include a line with the token name and password, in the form *token:password*. For example:

   ```
   internal:secret
   ```

   For the NSS software crypto module (the default software database), the token is always called **internal**.

   The password file should be owned by the Admin Server user and set to read-only by the Admin Server user, with no access to any other user (mode **0400**).

   NOTE

   To find out what the Admin Server user ID is, run **grep** in the Admin Server configuration directory:

   ```
   cd /etc/dirsrv/admin-serv

   grep \^User console.conf
   ```

3. In the **/etc/dirsrv/admin-serv** directory, edit the **nss.conf** file to point to the location of the new password file.

   ```
   #  Pass Phrase Dialog:
   #  Configure the pass phrase gathering process.
   #  The filtering dialog program (`builtin' is a internal
   #  terminal dialog) has to provide the pass phrase on stdout.
   NSSPassPhraseDialog  file://etc/dirsrv/admin-serv/password.conf
   ```

4. Restart the Admin Server. For example:

   ```
   service dirsrv-admin restart
   ```

After TLS/SSL is enabled, then the Admin Server can only be connected to using HTTPS. All of the previous HTTP (standard) URLs for connecting to the Admin Server and its services no longer work. This is true whether connecting to the Admin Server using the Console or using a web browser.

E.2.10. Changing Directory Server Settings

The Admin Server stored information about the Directory Server *Configuration Directory* (which stores the instance configuration information) and the Directory Server *User Directory* (which stores the actual directory entries). These can be the same directory instance, but they do not have to be. The settings for both of those databases can be edited in the Admin Server configuration so that it communicates with a different Directory Server instance.

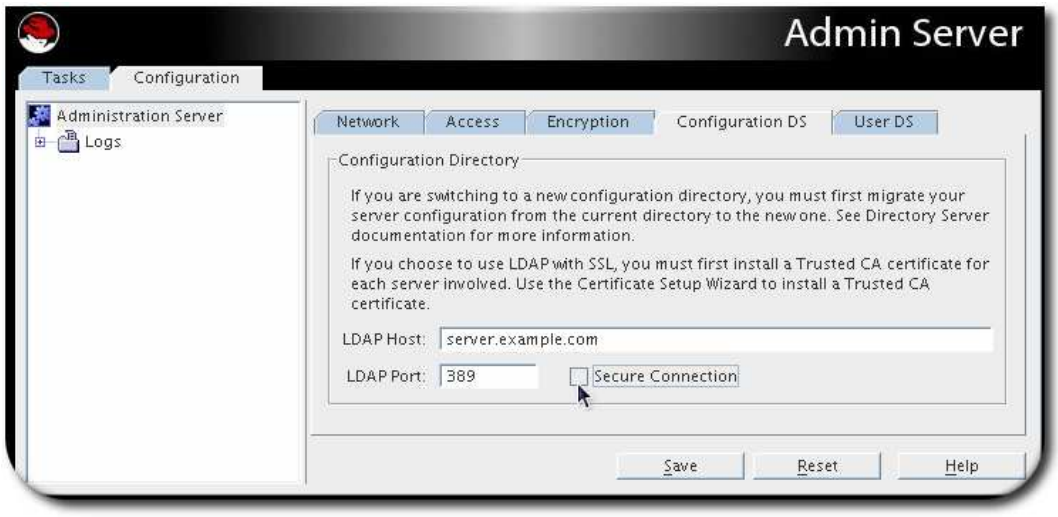
E.2.10.1. Changing the Configuration Directory Host or Port

Configuration data are stored under **o=NetscapeRoot** in the Configuration Directory. The configuration database contains server settings such as network topology information and server instance entries. When server configuration changes are stored in the configuration directory subtree.

WARNING

Changing the Directory Server host name or port number impacts the rest of the servers in the server group. Changing a setting here means the same change must be made for every server in the server group.

1. Open the Admin Server management window.
2. Click the **Configuration** tab.
3. Click the **Configuration DS** tab.
4. Set the Configuration Directory Server connection information.



   - The **LDAP Host** is the host name, IPv4, or IPv6 address of the Configuration Directory Server machine.
   - The **LDAP Port** is the port number to use for the Directory Server instance. The regular LDAP port is **389**; the default LDAPS (secure) port number is **636**.
   - Check the **Secure Connection** check box to use the secure port. Before checking this box, make sure that the Configuration Directory Server has enabled SSL.
5. Click **Save**.

E.2.10.2. Changing the User Directory Host or Port

The user directory is used for authentication, user management, and access control. It stores all user and group data, account data, group lists, and access control instructions (ACIs).

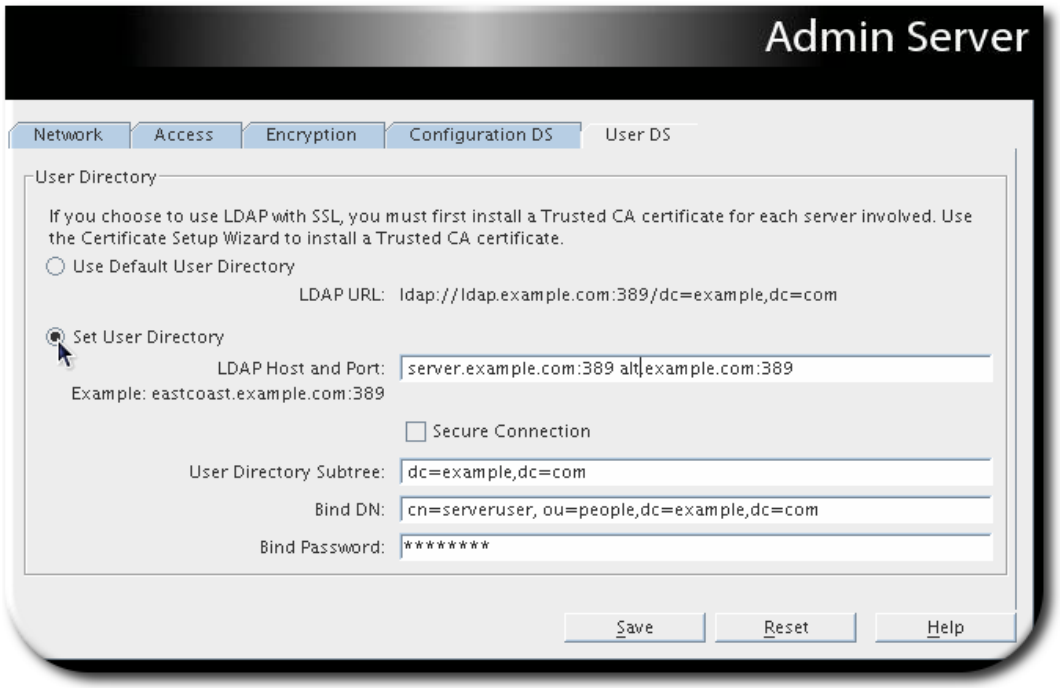
There can be multiple user directories in a single deployment because using multiple user directories enhances overall performance for organizations which are geographically spread out, which have high usage, or have discrete divisions which benefit from individual directories.

Admin Server can be configured to authenticate users against multiple user directories.

To change the information for the user directory:

1. Open the Admin Server management window.
2. Click the **Configuration** tab.
3. Click the **User DS** tab.

4. Set the User Directory Server connection information.
5. Edit the user directory information.



The **Use Default User Directory** radio button uses the default user directory associated with the domain. To use multiple Directory Server instances or to use a different instance, select the **Set User Directory** radio button and set the required information:

- The **LDAP Host and Port** field specifies the location of the user directory instance, using the format *hostname:port* or *ip_address:port*, with an IPv4 or IPv6 address.

  It is possible to configure multiple locations for the user directory for authentication and other directory functions; separate each location with a space. For example:

  ```
  server.example.com:389 alt.example.com:389
  ```

  NOTE

  If more than one location is given in the **LDAP Host and Port** field, the settings for the remaining fields will apply to all of those instances.

- Check the **Secure Connection** box to use SSL to connect to the user directory. *Only* select this if the Directory Server is already configured to use SSL.
- Give the **User Directory Subtree**. For example:

  ```
  dc=example,dc=com
  ```

  Every location listed in the **LDAP Host and Port** field must contain that subtree and the subtree must contain the user information.

- Optionally, enter the **Bind DN** and **Bind Password** for the user which connects to the user directory.

6. Click **Save**.

APPENDIX F. USING ADMIN EXPRESS

F.1. MANAGING SERVERS IN ADMIN EXPRESS

Admin Express provides a quick, simple web-based gateway to do basic management of servers. There are three tasks that can be performed through Admin Express:

- Stopping and starting the server
- Checking the server access, error, and audit logs
- Monitoring the progress and information for replication between Directory Servers

F.1.1. Opening Admin Express

The Admin Server services pages URL is the Admin Server host (host name, IPv4 address, or IPv6 address) and port. For example:

```
http://ldap.example.com:9830/
```

The Admin Express page is always available at that URL.

NOTE

If SSL/TLS is enabled on the Admin Server, then the URL must use the prefix **https** with the same port number. The standard HTTP URLs will not work.

```
https://ldap.example.com:9830/
```

F.1.2. Starting and Stopping Servers

On the main Admin Express page, there are buttons to turn servers off and on.

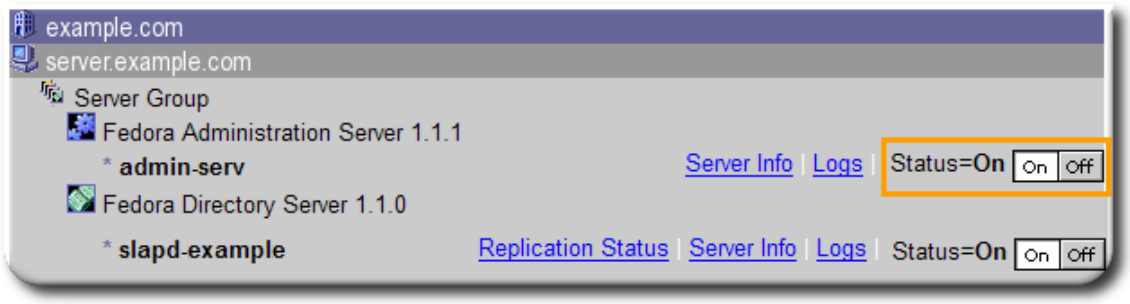


Figure F.1. Stopping and Stopping Servers

IMPORTANT

If either the Admin Server or the Configuration Directory Server is turned off through the Admin Express page, then it must be restarted through the command line, not through the Admin Express **On**/**Off** buttons because Admin Express requires access to both the Admin Server and Configuration Directory Server in order to function.

Other Directory Server instances can be safely stopped and restarted through Admin Express.

F.1.3. Viewing Server Logs

Admin Express can show and search the access and error logs for Directory Server and Admin Server and the audit logs for the Directory Server.

1. In the Admin Express page, click the **Logs** link by the server name.
2. Select which log type to view, how many lines to return, and any string to search for, and click **OK**.

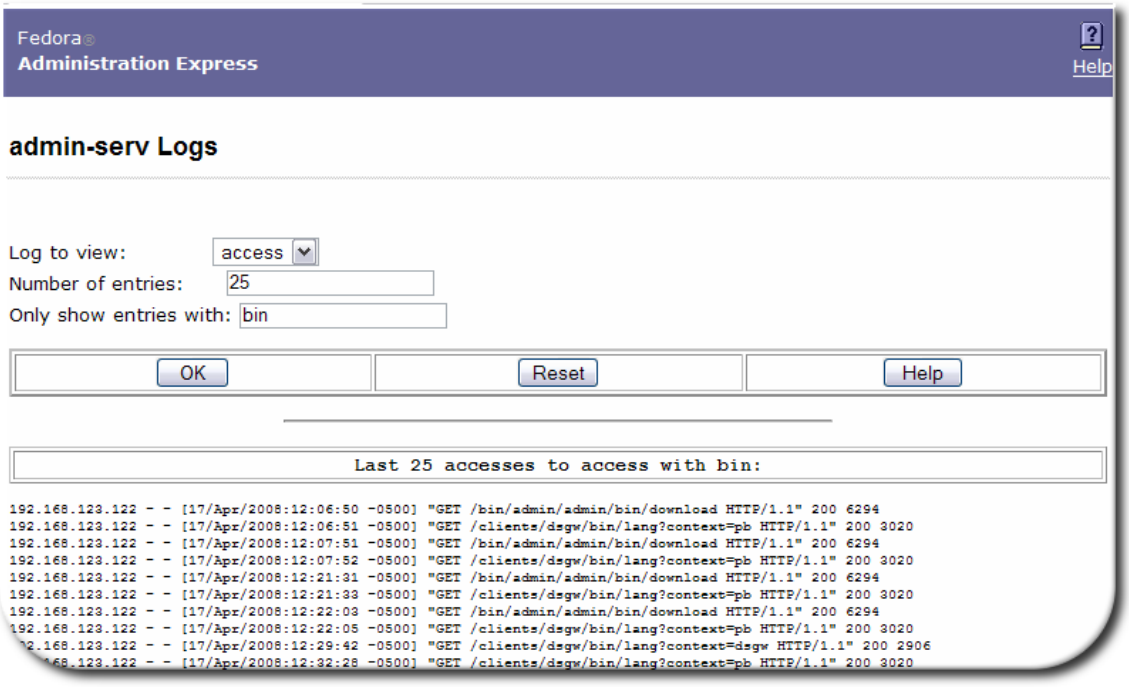


Figure F.2. Checking Logs

F.1.4. Viewing Server Information

The **Server Info** link on the Admin Express page opens a page with the basic description of the server instance, such as the build number, installation date, and server port number. This is the same information displayed in the Console when an instance is selected.

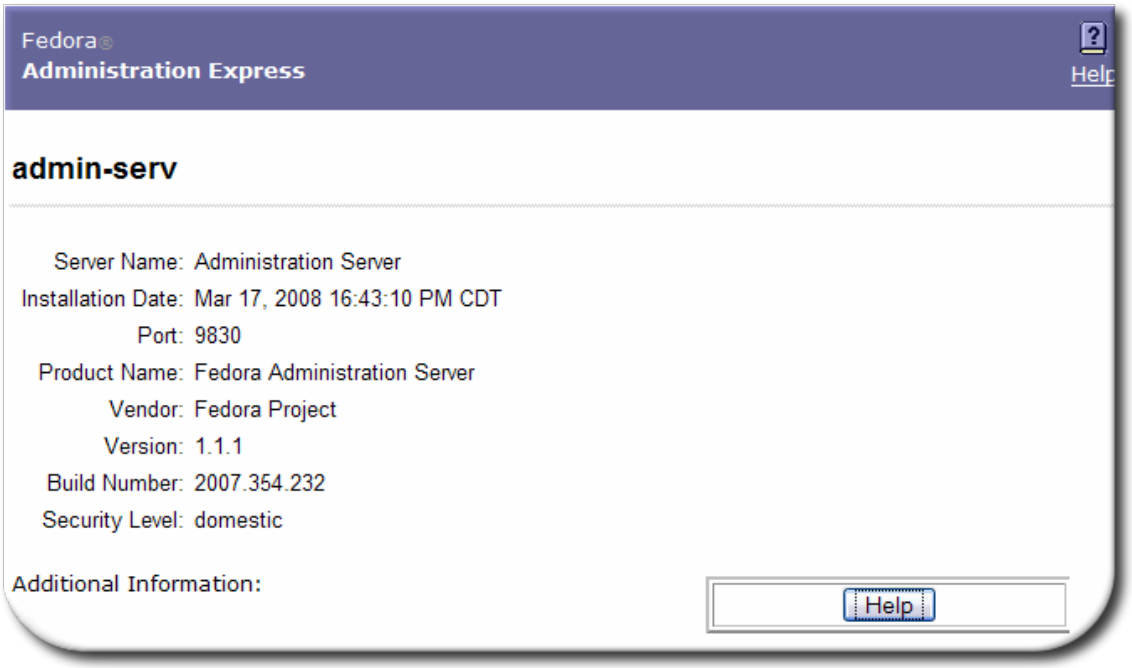


Figure F.3. Checking Server Information

The Directory Server information is located in the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file; the Admin Server information is located in **.conf** files in the **/etc/dirsrv/admin-serv** directory.

F.2. CONFIGURING ADMIN EXPRESS

Admin Express can be edited for the page appearance, but most functionality is controlled through the web server or the Admin Server configuration and should be edited through those servers, not by editing the configuration files directly.

F.2.1. Admin Express File Locations

The directories for all of the Admin Express configuration files are listed in Table F.1, "Admin Express File Directories"; the specific files are described in each section describing the different Admin Express page configurations.

Table F.1. Admin Express File Directories

Directory	Description
/etc/dirsrv/admin-serv	Contains the **local.conf**, **httpd.conf**, and other configuration files which define the Admin Server and configure the web server.
/usr/share/dirsrv/html/	Contains the HTML files and graphics used for the Admin Express appearance.

F.2.2. Admin Express Configuration Files

The behavior for Admin Express is mostly set through the web server configuration and should not be edited. The other Admin Express configuration is set through directives which insert data or form fields.

There is not cascading style sheet (CSS) file to centralize the formatting for pages in Admin Express. All formatting is done inline with the tags or through **<style>** tags in the page head. For information on editing inline tags, see http://directory.fedoraproject.org/docs/389ds/administration/htmlediting.html.

F.2.2.1. Files for the Admin Server Welcome Page

The configuration files for the introductory page for Admin Express is located in the **/etc/dirsrv/admin-serv** directory. One file sets the formatting, copyright text, and some web application text, **admserv.html**.

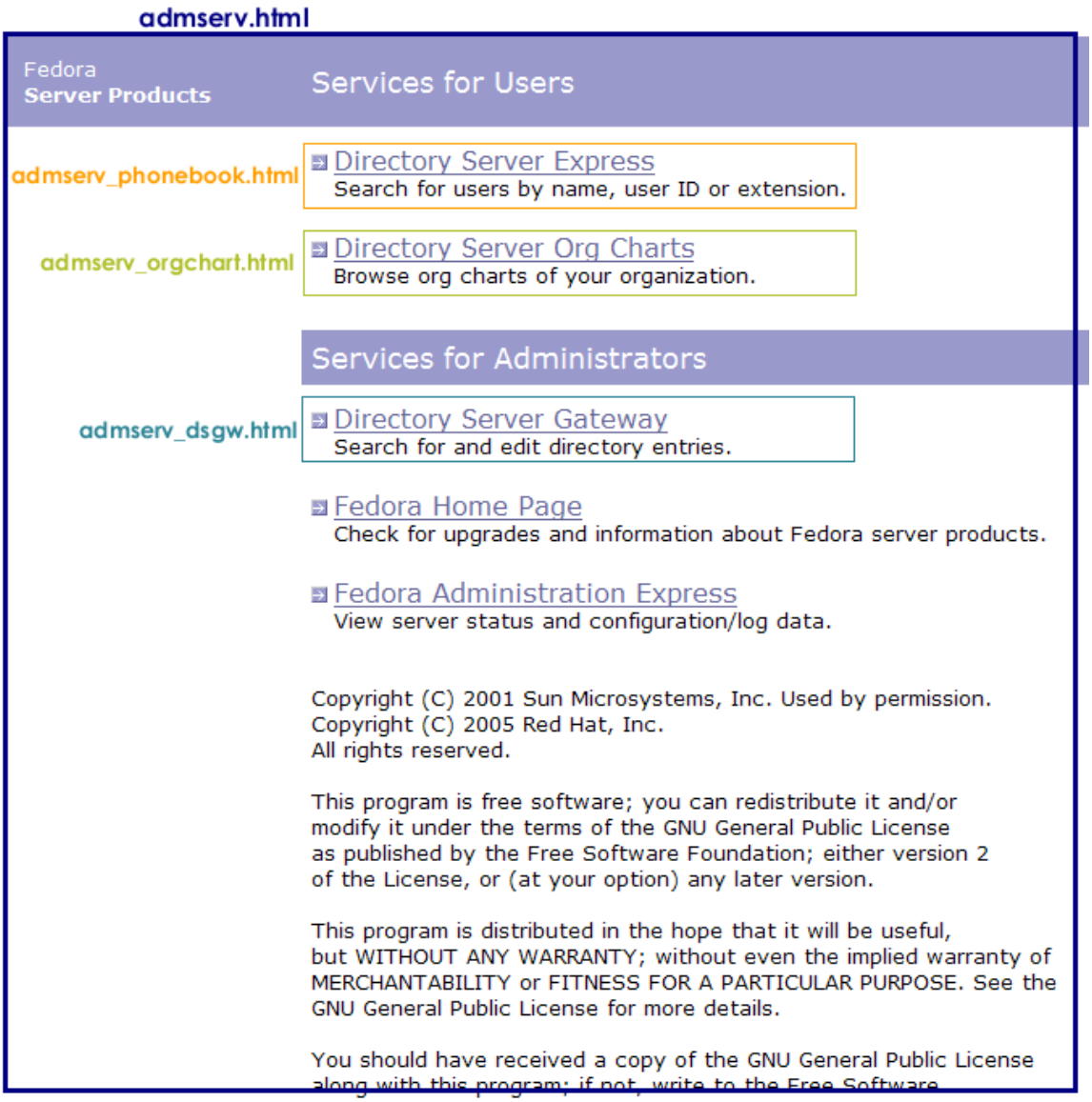


Figure F.4. Intro Page Elements

All of the formatting for the page is set inline. The text files are inserted using the **INCLUDEIFEXISTS** directive.

```
<tr valign="TOP">
  <td> </td>
  <td bgcolor="#9999cc" colspan="4"> <font color="white" size="+1"><font face="Verdana, sans-serif">Services
    for Administrators</font></font></td>
  <td> </td>
</tr>
<tr valign="TOP">
  <td> </td>
  <td colspan="4">
    <table border="0" cellspacing="0" cellpadding="0">
      <tr valign="TOP">
        <td><img src="/icons/spacer.gif" width="6" height="6"></td>
        <td></td>
      </tr>
<!-- INCLUDEIFEXISTS admserv_dsgw.html -->
```

The text files themselves have inline formatting for the inserted table rows.

F.2.2.2. Files for the Replication Status Appearance

There are two pages for monitoring the replication status. The first is for the configuration page, which requires two files:

- The body of the page, **/usr/share/dirsrv/html/monreplication.html**

- The heading of the page, **/usr/share/dirsrv/html/htmladmin.html**

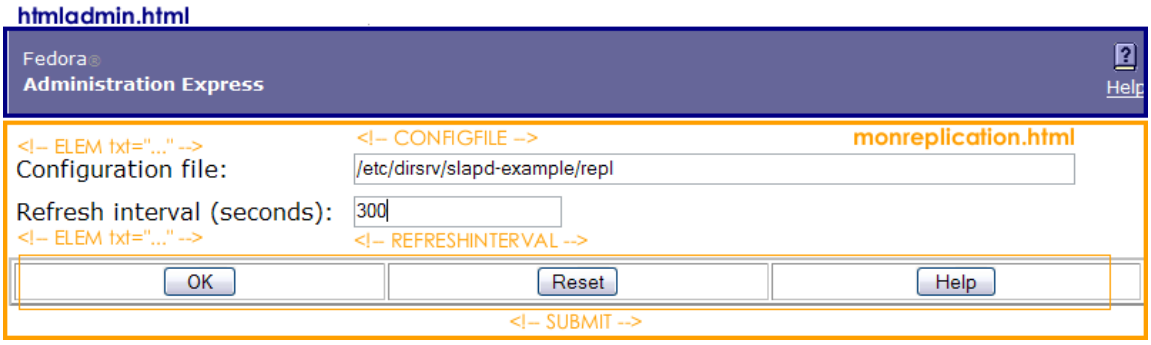


Figure F.5. Monitoring Replication Setup Page Elements

The **Replication Status** page uses two script-related configuration files:

- The body of the page, which is configured in the replication monitoring script, **/usr/bin/repl-monitor.pl**
- Optionally, the configuration file for the replication monitoring, which can configure the time lag colors with the **[colors]** section
- The heading of the page, **/usr/share/dirsrv/html/htmladmin.html**

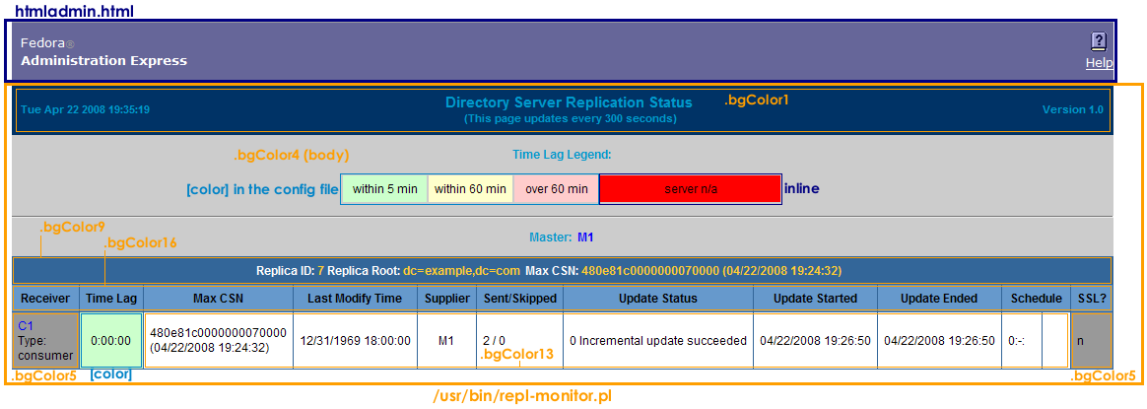


Figure F.6. Monitoring Replication View Page Elements

The text for the table headings, labels, and page sections are set in the Perl script. For example:

```
#Print the header of consumer
   print "\n<tr class=bgColor16>\n";
   print "<th nowrap>Receiver</th>\n";
   print "<th nowrap>Time Lag</th>\n";
   print "<th nowrap>Max CSN</th>\n";
....
   print "</tr>\n";
```

The styles for the **Replication Status** page are printed in the Perl script in the <style> tag in the HTML header. Many of the classes are the same as those in the **style.css** for the other web applications. These can be edited in the Perl script or by uncommenting the stylesheet reference and supplying a CSS file. For example:

```
 # print the HTML header

   print "Content-type: text/html\n\n";
   print "<!DOCTYPE HTML PUBLIC \"-//W3C//DTD HTML 3.2//EN\"><html>\n";
   print "<head><title>Replication Status</title>\n";
   # print "<link type=text/css rel=stylesheet href=\"master-style.css\">\n";
   print "<style text/css>\n";
   print "Body, p, table, td, ul, li {color: #000000; font-family: Arial, Helvetica, sans-serif; font-size: 12px;}\n";
   print "A {color:blue; text-decoration: none;}\n";
   print "BODY {font-family: Arial, Helvetica, sans-serif}\n";
   print "P {font-family: Arial, Helvetica, sans-serif}\n";
   print "TH {font-weight: bold; font-family: Arial, Helvetica, sans-serif}\n";
   print "TD {font-family: Arial, Helvetica, sans-serif}\n";
   print ".bgColor1  {background-color: #003366;}\n";
```

```
print ".bgColor4  {background-color: #cccccc;}\n";
print ".bgColor5  {background-color: #999999;}\n";
print ".bgColor9  {background-color: #336699;}\n";
print ".bgColor13 {background-color: #ffffff;}\n";
print ".bgColor16 {background-color: #6699cc;}\n";
print ".text8  {color: #0099cc; font-size: 11px; font-weight: bold;}\n";
print ".text28 {color: #ffcc33; font-size: 12px; font-weight: bold;}\n";
print ".areatitle {font-weight: bold; color: #ffffff; font-family: Arial, Helvetica, sans-serif}\n";
print ".page-title {font-weight: bold; font-size: larger; font-family: Arial, Helvetica, sans-serif}\n";
print ".page-subtitle {font-weight: bold; font-family: Arial, Helvetica, sans-serif}\n";

print "</style></head>\n<body class=bgColor4>\n";
```

F.2.2.3. Files for the Server Information Page

There are two files formatting the server information page:

- The body of the page, **/usr/share/dirsrv/html/viewdata.html**
- The heading of the page, **/usr/share/dirsrv/html/htmladmin.html**

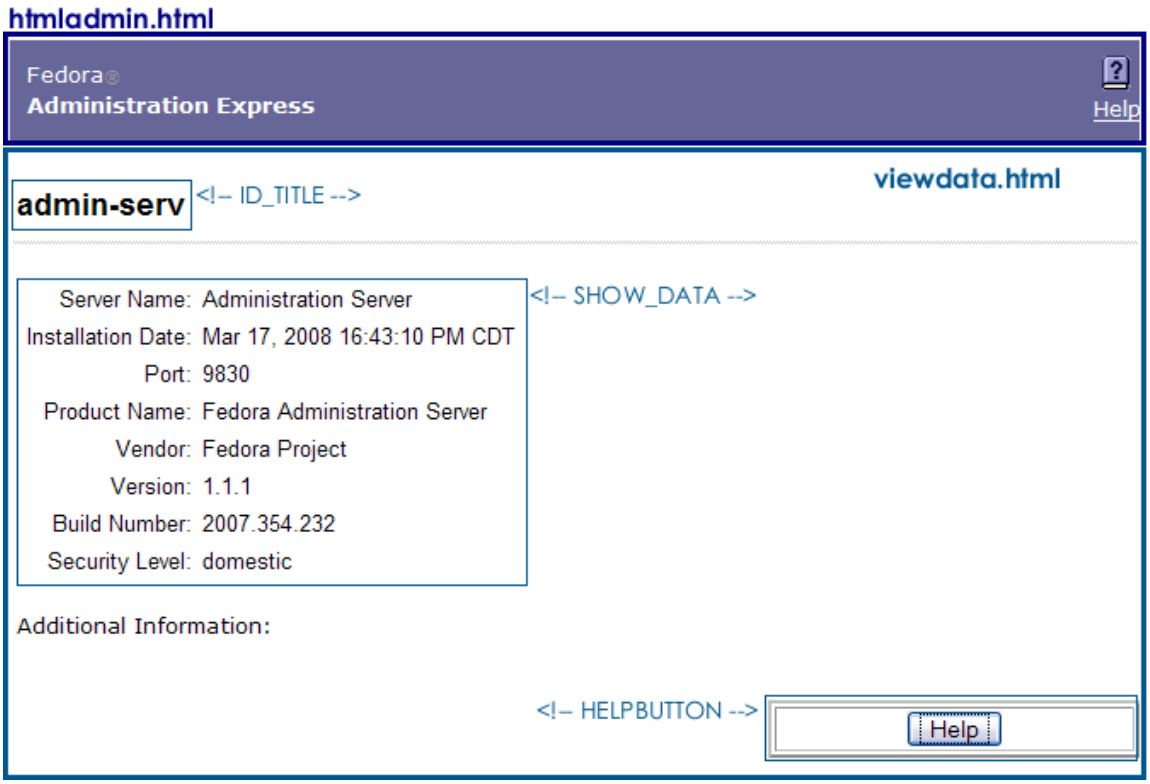


Figure F.7. Server Information Page Elements

The **viewdata.html** file is very simple, using only the two directives to insert the server data, plus other directives to insert other information. For the Admin Server, the **SHOW_DATA** directive takes the information from the **/etc/dirsrv/admin-serv/local.conf** file. For the Directory Server, it takes the data from the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file. The **ID_TITLE** is the name of the server instance.

```
<body text="#000000" bgcolor="#FFFFFF" link="#666699" vlink="#666699" alink="#333366">

<br>
<table BORDER=0 CELLSPACING=2 CELLPADDING=2 WIDTH="100%">
<!-- ID_TITLE -->
<p>
<!-- SHOW_DATA -->
<p>
<font face="PrimaSans BT, Verdana, sans-serif"><font size=-1>Additional Information:</font></font>
<p>
<!-- CHECK_UPGRADE -->
<p>
<!-- SHOW_URL -->
</table>
```

```
<!-- HELPBUTTON -->

</body>
```

F.2.2.4. Files for the Server Logs Page

There are two files formatting the server logs page:

- The body of the page, **/usr/share/dirsrv/html/viewlog.html**
- The heading of the page, **/usr/share/dirsrv/html/htmladmin.html**

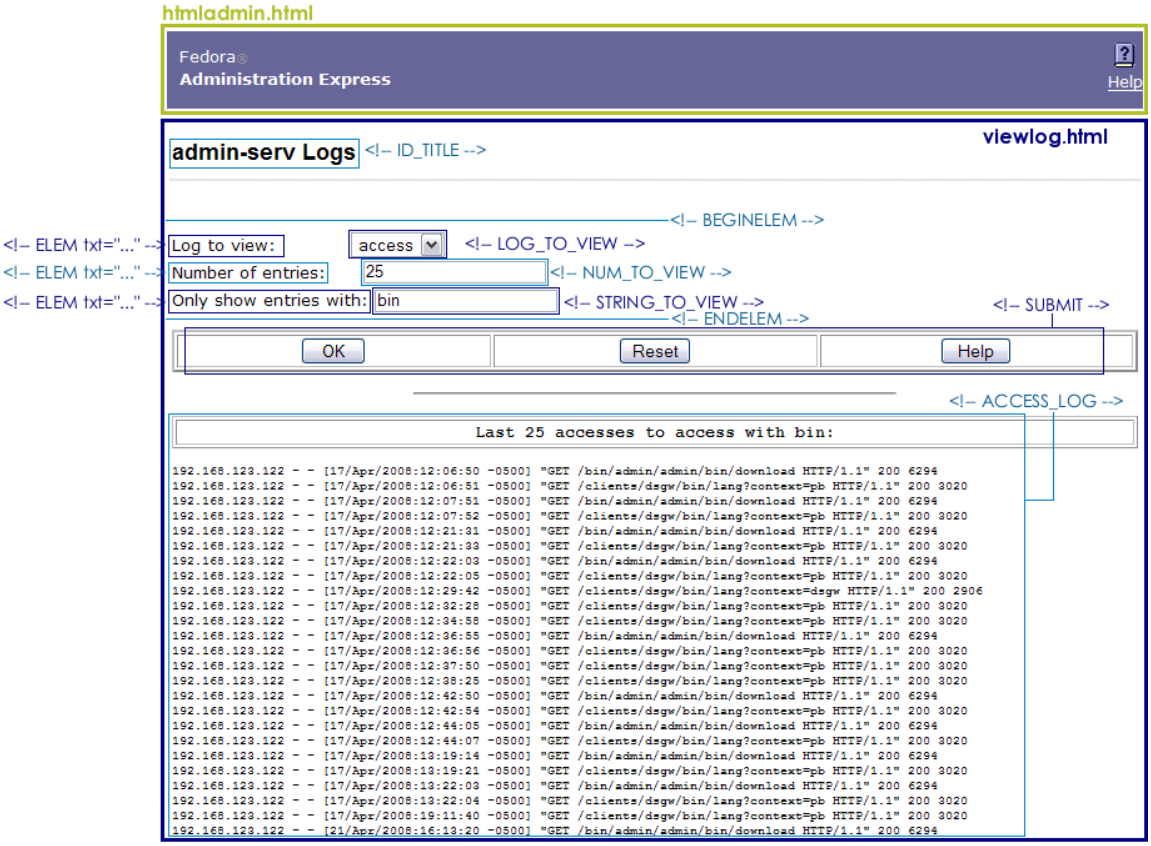


Figure F.8. Log View Page Elements

The page information is set through the inserted directives. The server instance name is set in the **ID_TITLE** directive. The log is displayed through the **ACCESS_LOG** directives. The form at the top is formatted with directive pairs, one which sets the descriptive text and the other inserting the field type. For example, this sets the log type menu:

```
<form method=GET action=ViewLog>
<font face="PrimaSans BT, Verdana, sans-serif"><font size=-1>
<!-- BEGINELEM -->
<!-- ELEM txt="Log to view:           " -->
<!-- LOG_TO_VIEW -->
....
<!-- SUBMIT -->
</font></font>
</form>
```

F.2.3. Admin Express Directives

The Admin Express directives are HTML comments that are interpreted by the CGI scripts; these directives are used to set form fields and to pull data from the server configuration and log files.

Table F.2. Admin Express Directives

Directive	Description	Example

Directive	Description	Example
ACCESS_LOG	Inserts the server log file.	<!-- ACCESS_LOG -->
ADMURL		<!-- ADMURL -->
BEGINELEM	Marks the opening of form input elements. This is always paired with **ENDELEM**.	<!-- BEGINELEM -->
CHECK_UPGRADE		<!-- CHECK_UPGRADE -->
ELEM	Inserts a text element. This has one argument, **txt=**, which defines the text to use.	<!-- ELEM txt="Field name here: " -->
ELEMADD	Inserts a text element. This has one argument, **txt=**, which defines the text to use.	<!-- ELEMADD txt="Field name here: " -->
ENDELEM	Marks the ending of form input elements. This is always paired with **BEGINELEM**.	<!-- ENDELEM -->
HELP_BUTTON	Inserts a button to open context-specific help.	<!-- HELP_BUTTON -->
HELPLINK	Inserts a link to the general Admin Express help file.	<!-- HELPLINK -->
HIDDEN_ID		<!-- HIDDEN_ID -->
ID_TITLE	Inserts the name of the server instance, such as **admin-serv** or **example** (if the Directory Server instance name is **slapd-example**)	<!-- ID_TITLE -->
INCLUDEIFEXISTS	Inserts the contents of the HTML file. The inserted file should include both the text and any HTML markup.	<!-- INCLUDEIFEXISTS "file.html" -->
LOG_TO_VIEW	Inserts a drop-down menu with the types of logs available to view.	<!-- LOG_TO_VIEW -->
NUM_TO_VIEW	Inserts a form field to set the number of lines to return.	<!-- NUM_TO_VIEW -->
REFRESHINTERVAL	Inserts a form field to set the refresh interval (in seconds) for replication monitoring.	<!-- REFRESHINTERVAL -->
SERVHOST		<!-- SERVHOST -->
SERVPORT		<!-- SERVPORT -->
SHOW_DATA	Inserts the server data from the configuration file, including the port number, installation date, and build number.	<!-- SHOW_DATA -->
SHOW_URL		<!-- SHOW_URL -->
SITEROOT		<!-- SITEROOT -->

Directive	Description	Example
STRING_TO_VIEW	Inserts a form field to use to set the search string for the logs.	<!-- STRING_TO_VIEW -->
SUBMIT	Inserts a three-button set: to save or submit the form; to reset the form; and to open a help topic.	<!-- SUBMIT -->

APPENDIX G. USING THE CONSOLE

G.1. OVERVIEW OF THE DIRECTORY SERVER CONSOLE

Red Hat Management Console is the user interface to manage Red Hat Directory Server and Admin Server configuration and directory information. There is a single main Console window which administers the servers (collected and identified in *administration domains*). The main Console allows you to open server-specific Consoles to manage the settings and information in individual instances.

This chapter provides an overview of how the Console interacts with the Directory Server and Admin Server and walks through the Console windows and options.

G.1.1. How the Console, Directory Server, and Admin Server Work Together

The Red Hat Console is an independent Java application which works in conjunction with instances of Red Hat Directory Server and Admin Server. Most server management functions are carried out in server-specific console windows for the Directory Server and Admin Server. Red Hat Console is part of a system that manages Red Hat Directory Server instances and the Admin Server and, therefore, information in the directory. Although Red Hat Directory Server, Red Hat Management Console, and Red Hat Admin Server work tightly with one another, each plays a specific role in managing servers, applications, and users.

Red Hat Management Console is the front-end management application for Red Hat Directory Server. It finds all servers and applications registered in the configuration directory, displays them in a graphical interface, and can manage and configure them. The Main Console can also search for, create, and edit user and group entries in the user directory.

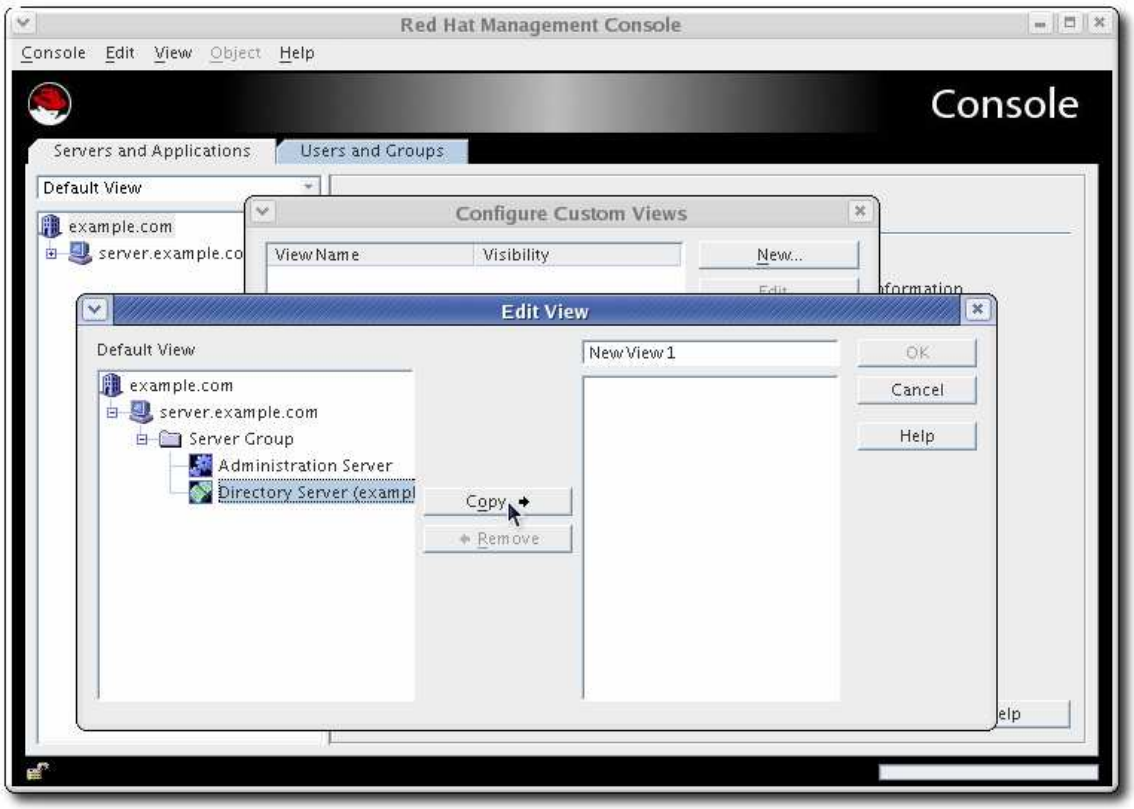


Figure G.1. The Red Hat Management Console Interface

When a user logs into Red Hat Management Console, the Console connects to the Admin Server over Hypertext Transfer Protocol (HTTP). The Admin Server receives requests to administer the different Directory Server instances and performs the changes to the configuration, such as changing a port number. When a request is sent to the Red Hat Management Console to add or edit user entries, the Console sends a Lightweight Directory Access Protocol (LDAP) message directly to Directory Server to update the user directory.

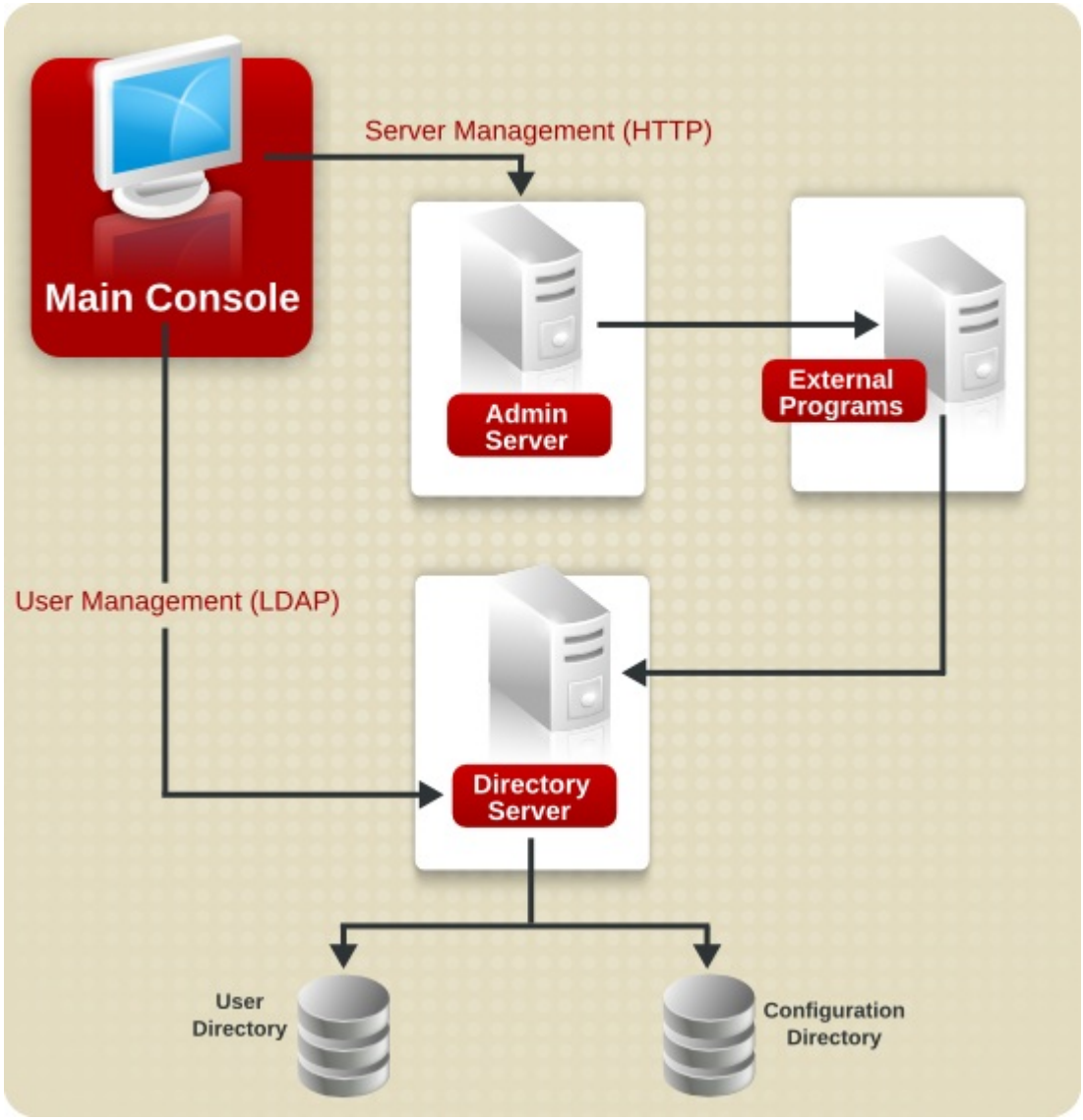


Figure G.2. Simple System Using Red Hat Management Console

Red Hat Directory Server stores server and application configuration settings as well as user information. Typically, application and server configuration information is stored in one subtree of Red Hat Directory Server while user and group entries are stored in another subtree. With a large enterprise, however, configuration and user information can be stored in separate *instances* of Directory Server (which can be on the same host machine or on two different host machines). Figure G.2, "Simple System Using Red Hat Management Console" illustrates a relatively simple Red Hat Directory Server system. As an enterprise grows and needs change, additional hosts and Directory and Admin Servers can be added to the administration domain in the Console, so that a single Console can manage multiple Directory and Admin Servers.

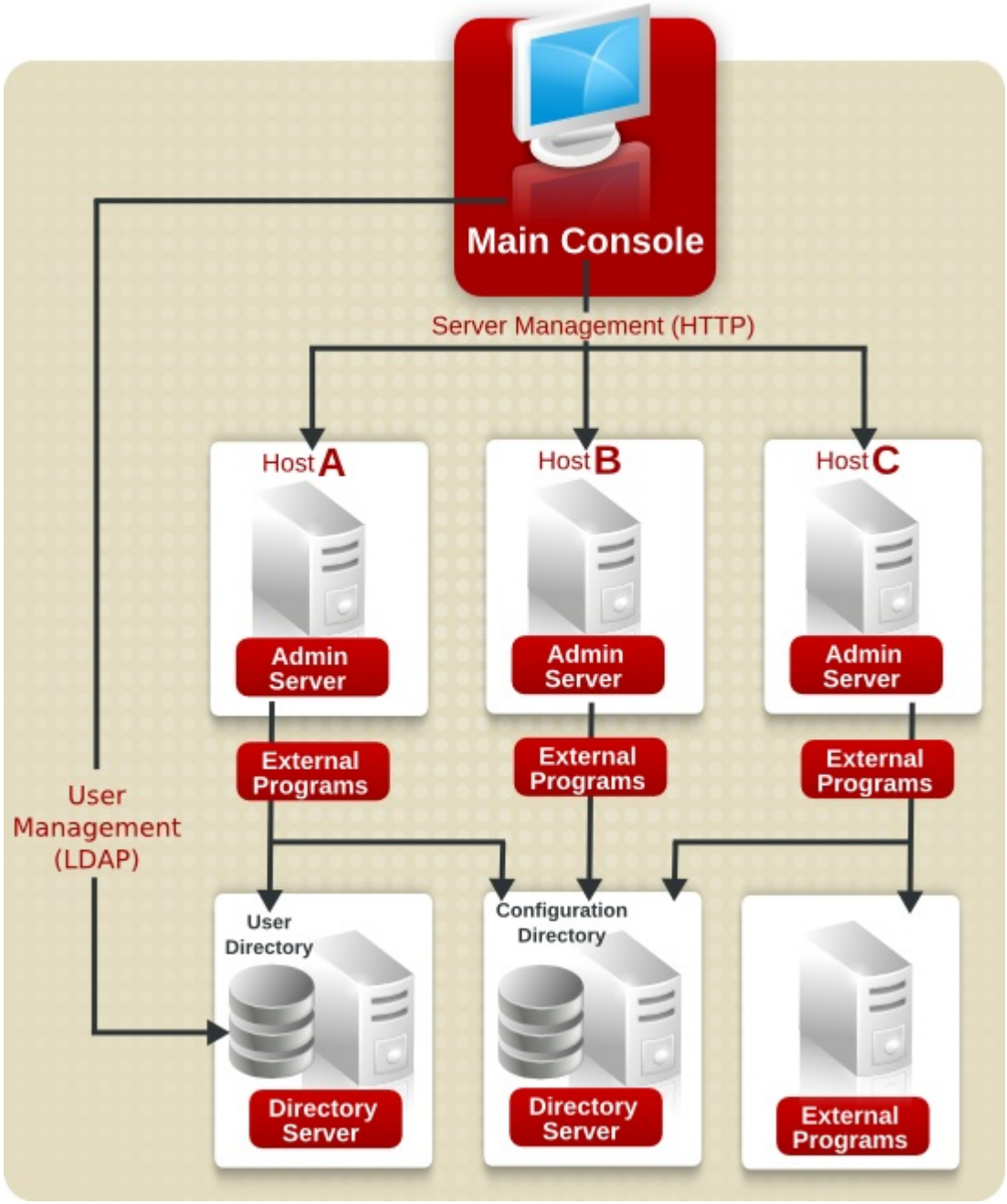


Figure G.3. A More Complex System

NOTE

When the terms *configuration directory* and *user directory* are used in this guide, they define where the configuration information and the user information is stored, regardless of whether that is in the subtrees of a single instance of Directory Server or in two separate instances of Directory Server.

G.1.2. Red Hat Management Console Menus

There are five menu items in the top menu the Console. The options for each of these menus varies depending on the Console window open (the main Console, Directory Server Console, or Admin Server Console) and the types of objects available in that server area.

Figure G.4. Main Console Menus

Table G.1. Console Menus

Menu	Description
Console	Manages the Console session, such as closing the window or exiting the session entirely. • For the main window, this menu also can be used to add and remove admin domain. • For the Directory Server Console, this allows people to log in as a different user. • For the Admin Server Console, it manages security issues, such as certificates and tokens.
Edit	Sets display preferences, for all three Consoles. For the Directory Server Console, this also provides ways to copy, paste, and delete directory entries or text.
View	Sets whether to display certain parts of the Console window, such as the top banner, menus, and side navigation panes. This also refreshes the current display. For the Directory Server Console, this menu also sets what parts of the directory or which databases to view.
Object	Provides available operations for the active object; this is the same as the right-click menu for the active area or entry. • For the main window, this menu simply opens or deletes a server instance. • For the Directory Server Console, this provides all of the configuration options for the directory entries, such as advanced property editors or creating new entries. • For the Admin Server Console, this opens a configuration editor, starts, and stops the server.
Help	Opens context-specific help for the current Console area.

G.1.3. Red Hat Management Console Tabs

There are two tabs in the main Console window:

- **Servers and Applications**, for managing the Directory Server and Admin Server instances
- **Users and Groups**, for searching for and creating user and group entries within the Directory Server

G.1.3.1. The Servers and Applications Tab

The **Servers and Applications** tab, by default, has a navigation tree on the left for viewing hosts and Directory and Admin Servers and a center information panel. To access the Directory Server instance, directory information, or

Admin Server, open the server resource listed in the navigation tree. The information for the server instance, such as the build number and port number,

The navigation tree displays the Red Hat Directory Server *topology*, a hierarchical representation of all the resources (such as servers and hosts), that are registered in a configuration directory.

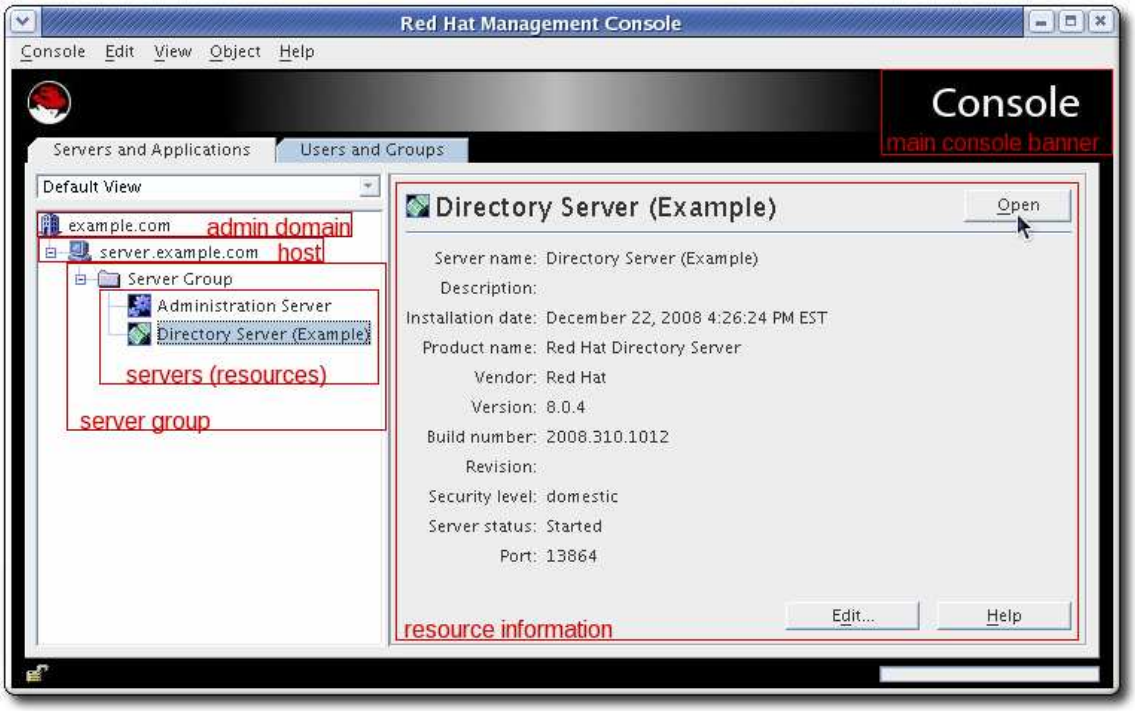


Figure G.5. The Servers and Applications Tab

The top of the topology is the *administration domain*. An administration domain is a collection of host systems and servers that share the same user directory. The server which hosts Directory Server or Admin Server instances belongs to the admin domain; that is the *host*.

A *server group* consists of all Directory Servers that are managed by a common Admin Server. A number of server groups can exist within an administration domain.

G.1.3.2. The Users and Groups Tab

The **Users and Groups** tab can search for user and group entries in any Directory Server administered by the Console. Any of the returned entries can be edited or deleted through this tab, assuming that the users has the proper access permissions. New entries can also be created through the **Users and Groups** tab.

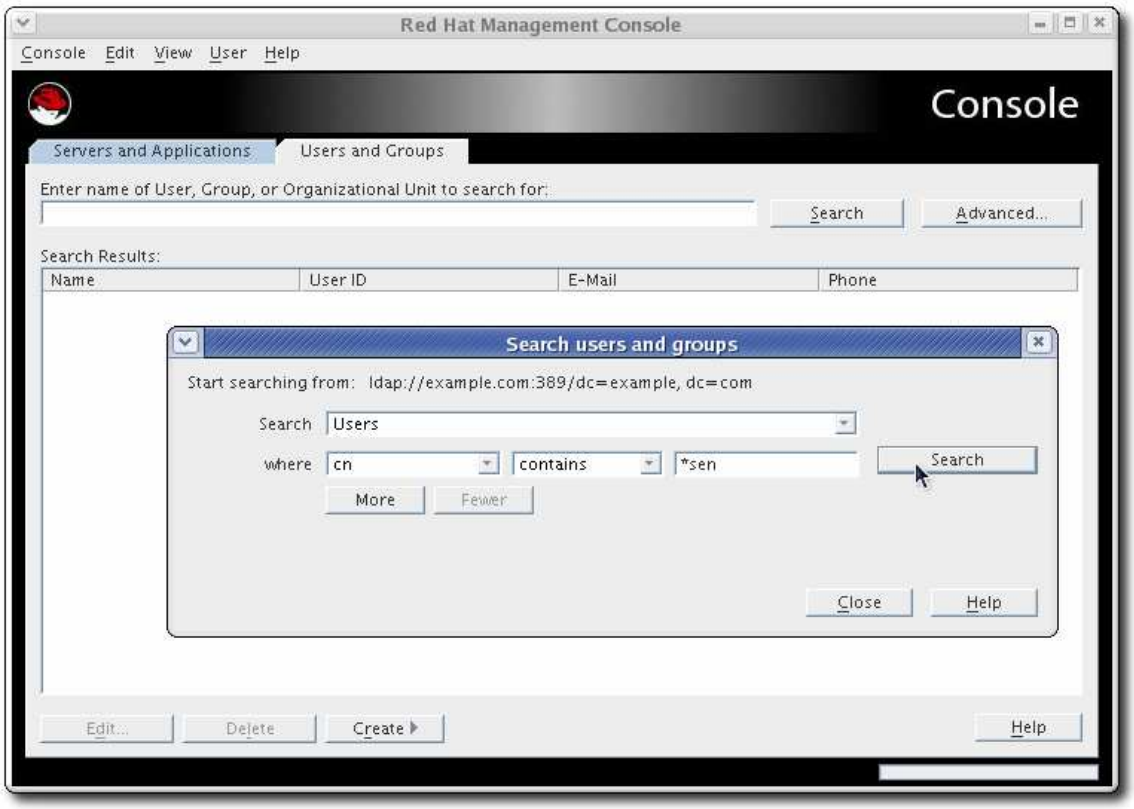


Figure G.6. The Users and Groups Tab

Switch the directory being searched or where the entries are added through the options in the **Users** menu, as described in Section G.4.1, "Searching for Users and Groups".

G.1.4. Server-Specific Consoles

The main Console can open into two server-specific windows to manage the Admin Server and Directory Server. These windows are opened by clicking the server name in the navigation area, and then clicking the **Open** button in the resources area.

G.1.4.1. The Directory Server Console

The Directory Server Console manages the specific Directory Server instance configuration, including the port number, SSL settings, and logging. The Directory Server Console also manages the directory information (entries) and directory operations like importing and exporting databases, creating suffixes, and extending the schema.

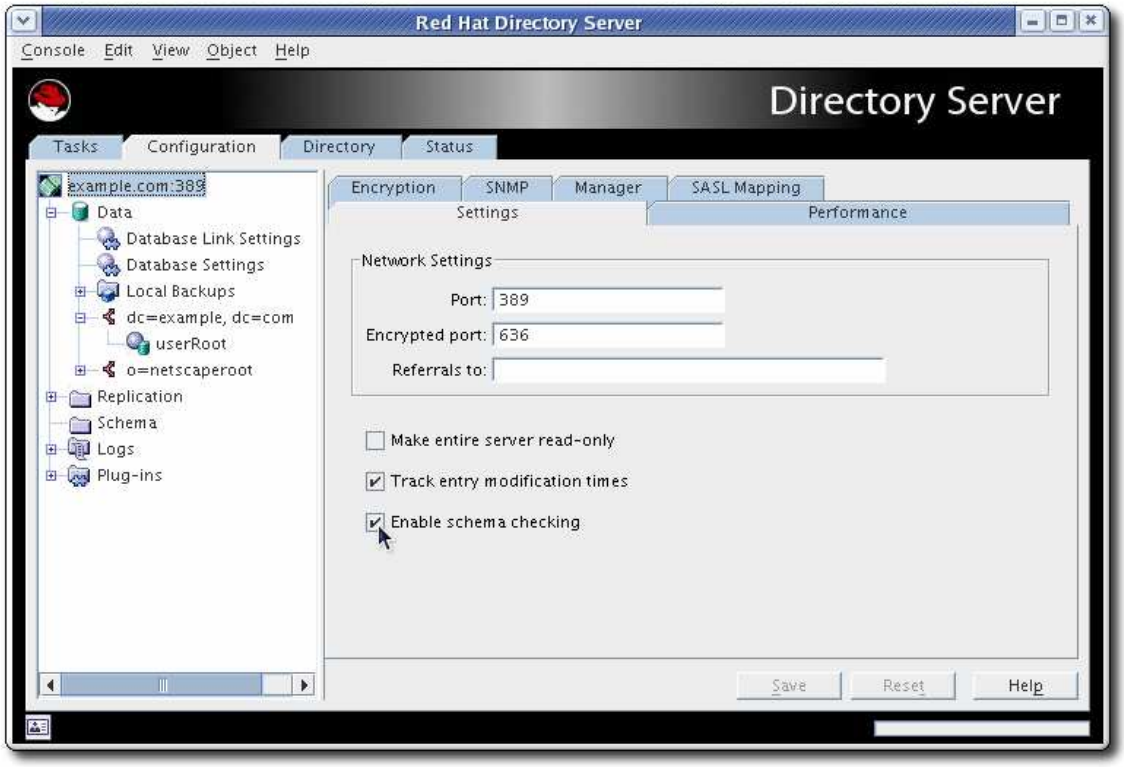


Figure G.7. The Directory Server Console

There are four tabs in the Directory Server Console:

- **Tasks**, which has shortcuts to common server operations, including starting and stopping the Directory Server instance, importing and exporting databases, and managing SSL certificates
- **Configuration**, which defines all of the server configuration settings, including SASL and SSL authentication, port numbers, schema, replication and synchronization, databases and suffixes, logging, and plug-ins
- **Directory**, which access and manages the directory information, including user entries and all group entries, including roles, classes of service, views, and groups
- **Status**, which monitors the server performance and displays the different monitoring and performance counters for the Directory Server and databases

Similar to the main Console, the Directory Server Console tabs have a navigation area on the left and a center panel that displays information about the active setting, entry, or database.

The procedures for using the Directory Server Console to manage the Directory Server configuration and directory entries is covered in the *Directory Server Administrator's Guide*.

G.1.4.2. The Admin Server Console

The Admin Server itself administers the configuration of other servers, especially the configuration and user directories for the server group. The Admin Server Console manages the Admin Server settings and the settings for these two Directory Server directories; whenever the settings are changed in the Directory Server configuration, the modifications must be carried into the Admin Server configuration for the server to properly manage those servers.

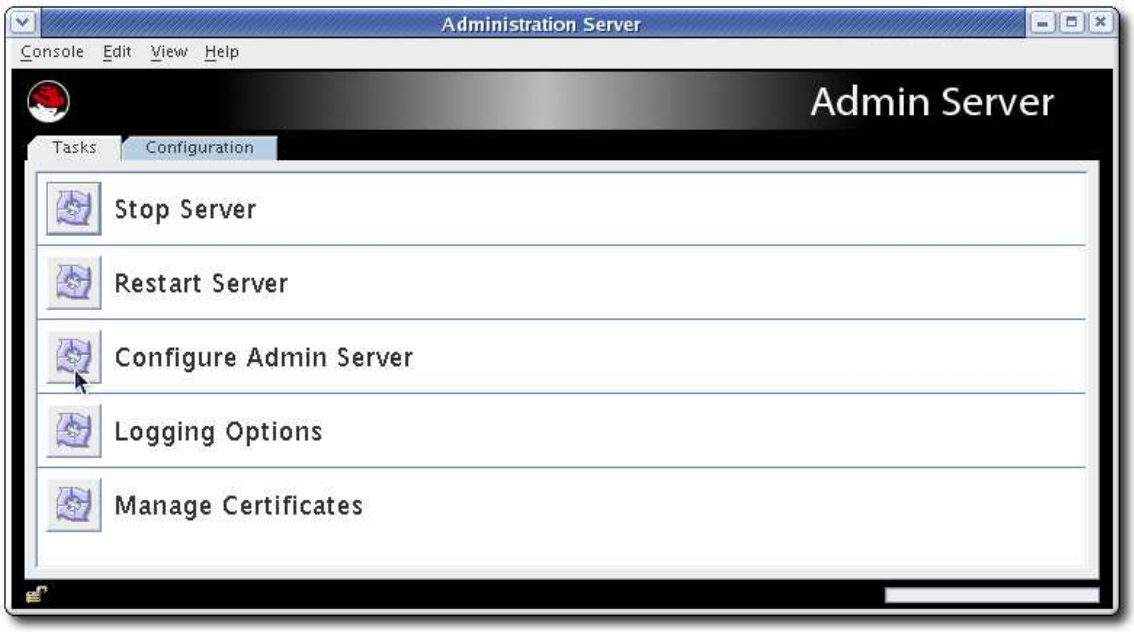


Figure G.8. The Admin Server Console

The Admin Server Console is simpler than the Directory Server Console, with only two tabs:

- **Tasks**, which has shortcuts to common server operations, including starting and stopping the Admin Server instance, setting up logging, and managing SSL certificates
- **Configuration**, which defines all of the Admin Server configuration settings, including SSL authentication, port numbers, and logging, as well as the Configuration Directory Server and User Directory Server settings which the Admin Server uses to connect to the directory services

The procedures for using the Admin Server Console to manage the Admin Server configuration and associated directory services is covered in the *Using the Admin Server* guide.

G.2. BASIC TASKS IN THE RED HAT CONSOLE

While most server management functions are carried out in server-specific console windows for the Red Hat Directory Server and Admin Server, the main Red Hat Console itself has some basic management functions, such as creating server instances, searching the directory, setting some access controls, and allowing some entry modifications.

This chapter covers basic tasks in the Red Hat Console, including installing the Console, creating and editing server instances, and configuring the Console appearance.

G.2.1. Installing the Console

The Red Hat Console package, **389-ds-console.noarch**, can be installed on Red Hat Enterprise Linux systems using tools like **yum**. For example:

```
yum install redhat-idm-console
```

The Red Hat Console package and also be downloaded through Red Hat Network and installed using package management tools such as **rpm** and **pkgadd**. For example:

```
rpm -ivh redhat-idm-console-1.0.0-22.el4idm.i386.rpm
```

G.2.2. Launching the Console

1. Run the **redhat-idm-console** command. For example:

   ```
   redhat-idm-console -a http://server.example.com:9830
   ```

   The different options for the **redhat-idm-console** command are listed in Table G.2, "Arguments for redhat-idm-console".

2. Enter the user name and password.

Also, enter or select the URL for the instance of Admin Server, if one was not passed with the command. The URL can be either the host name or the IP address of the Admin Server host. The Admin Server port number must be given, as well. The five most recent Admin Server URLs accessed are available as a drop-down menu option.

Table G.2. Arguments for redhat-idm-console

Argument	Description	Example
-a *adminURL*	Specifies a base URL for the instance of Admin Server to log into.	-a http://eastcoast.example.com:987
-f *fileName*	Writes errors and system messages to *fileName*.	-f system.out
-h	Prints out the help message for **redhat-idm-console**.	
-s	Specifies the directory instance to access, either by specifying the DN of the server instance entry (SIE) or the instance name, such as **slapd-example**.	-s slapd-example
-u	Gives the user DN to use to log into the Console.	-u "cn=Directory Manager"
-w	Gives the password to use to log into the Console.	-w secret
-w -	Reads the password from the standard output.	
-x *options*	Specifies extra options. There are three values for *extraOptions*: nowinpos, which puts the Console window in the upper left corner of the screen nologo, which keeps the splash screen from being displayed and only opens the login dialog javalaf, which uses the *Java look and feel* for the Console interface rather than the platform-specific styles To use multiple options, separate them with a comma.	-x nologo,nowinpos
-y *file*	Reads the password from the specified input file.	-y password.txt

G.2.3. Opening a Directory or Admin Server Window

The Red Hat Management Console is the avenue to access instance-specific management windows for the Directory Server and Admin Server. To open a console window for a specific server instance:

1. Open the Red Hat Console.

```
redhat-idm-console
```

2. Click the **Servers and Applications** tab, which lists all of the Directory Server and Admin Server instances within the configured Directory Server domain.

3. In the navigation tree, click a server to select it.

4. In the right-hand panel, click **Open**.

Alternatively, double-click the server icon in the navigation tree.

G.2.4. Changing the Console Appearance

The fonts used for different elements in the Console can be edited. The font settings and the location where the font profiles are stored can be customized. The default font settings can be restored easily.

This section also describes how to control other aspects of the appearance of the Console. For example, table columns can be easily rearranged. It is also possible to control which server instances are displayed (called a *navigation view*) which makes it easy to sort and find server instances.

Access control instructions can be applied to user interface elements, which is discussed in Section G.5, "Setting Access Controls".

- Section G.2.4.1, "Changing Profile Locations"
- Section G.2.4.2, "Restoring Default Font Settings"
- Section G.2.4.3, "Changing Console Fonts"
- Section G.2.4.4, "Reordering Table Columns"
- Section G.2.4.5, "Customizing the Main Window"

G.2.4.1. Changing Profile Locations

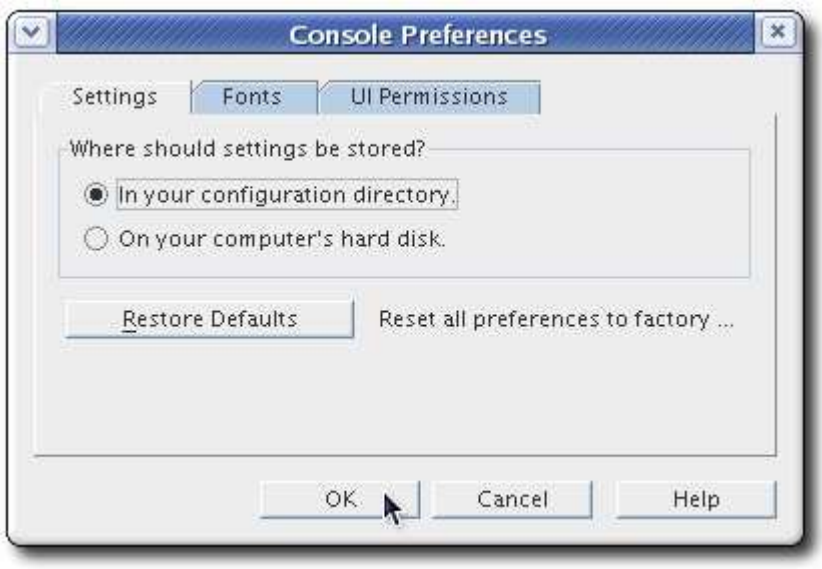
The Console formatting is stored in *profiles*. An entry's profiles can be stored locally, which means that they are only available at a specific workstation, or can be stored in the configuration directory, so they are accessible anywhere.

To set the profile location:

1. Click **Edit** in the top menu, and choose **Preferences**.

2. Click the **Settings** tab.

3. Select the radio button for the location to save the settings.



   - *In your configuration directory* means that the settings are stored in the Directory Server configuration, making them available no matter where you log into the Console.
   - *On your computer's hard disk* stores the setting profiles locally. This is mainly useful if you want specific, different settings used by default on different Consoles, such as a workstation and a laptop.

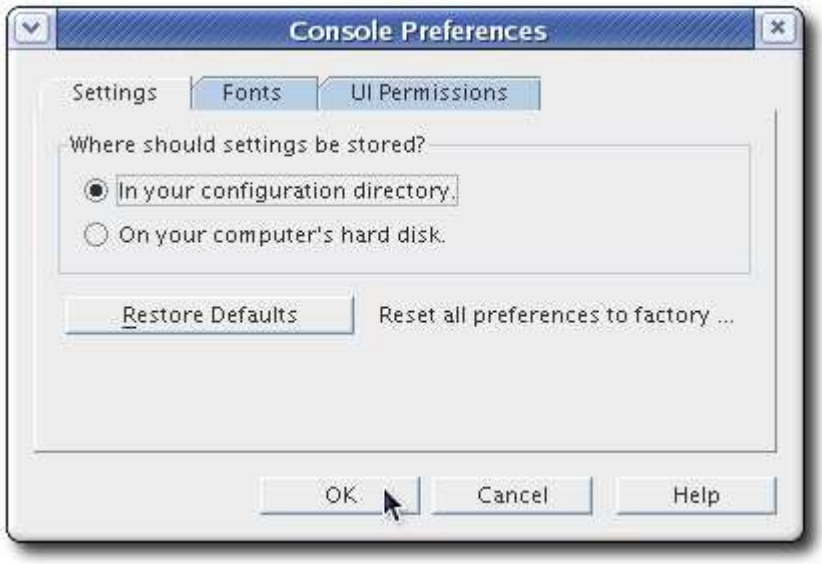
4. Click **OK**.

G.2.4.2. Restoring Default Font Settings

1. Click **Edit** in the top menu, and choose **Preferences**.

2. Click the **Settings** tab.

3. Click the **Restore Defaults** button to revert to the default display settings.



4. Click **OK**.

G.2.4.3. Changing Console Fonts

Different parts of the Console, such as table headings and regular text, have different font settings. The font settings are stored in *profiles*. The profiles define the font family, size, and formatting for every text element. There can be multiple font profiles available, and the font profiles can be private, such as settings for a specific user or group, or public, so that any user can access them.

The default profile can be edited without having to create new profiles.

To edit or create a font profile:

1. In the main Red Hat Management Console window, from the **Edit** menu, choose **Preferences**.

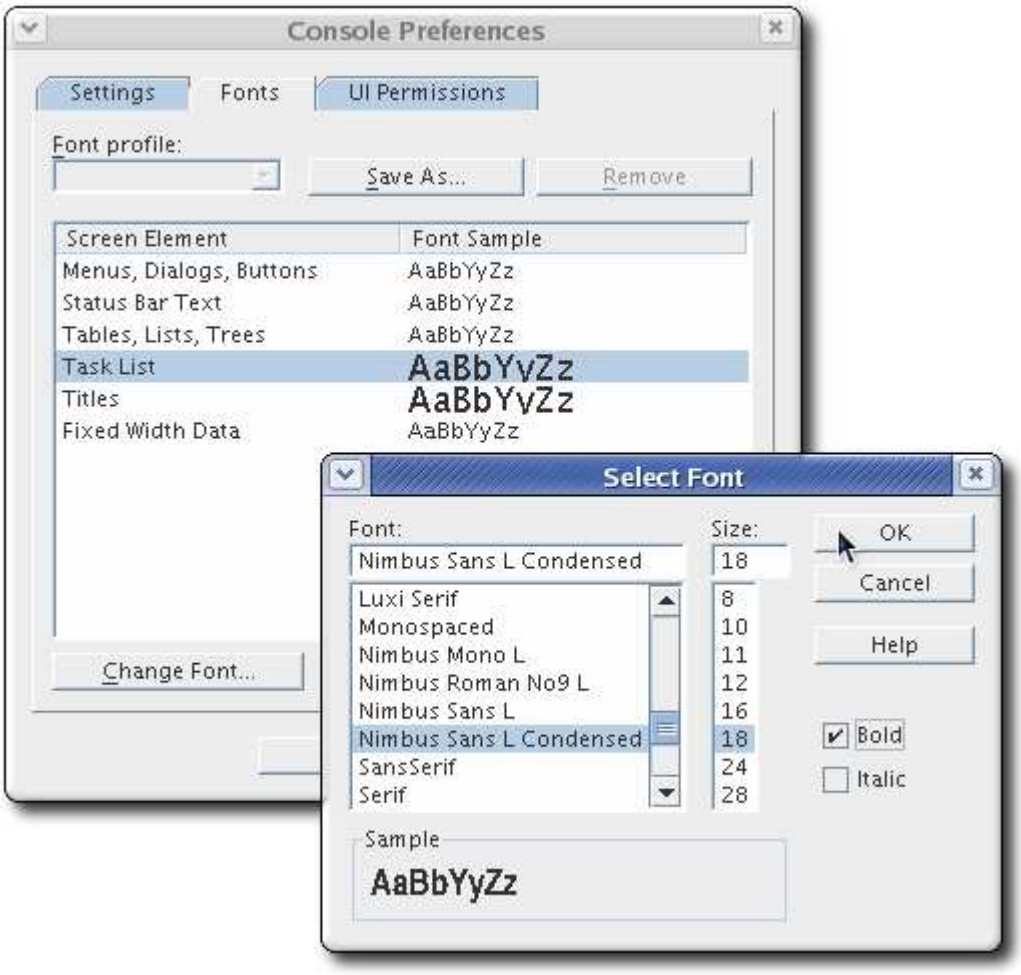
2. Click the **Fonts** tab.

3. To save the new settings as a new profile, click the **Save As** button, and fill in the profile name.

   To edit the default (or current) profile, simply begin editing the fonts.

4. In the **Screen Element** column, click a screen element to edit, then click the **Change Font** button.

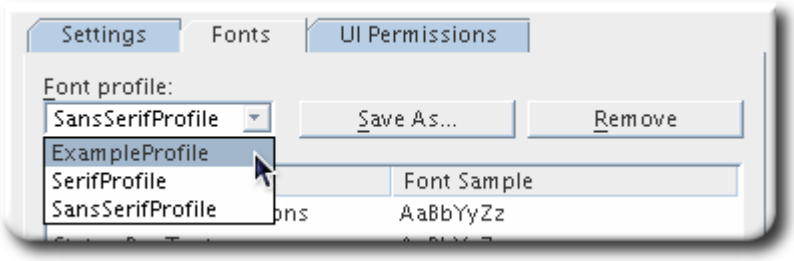
5. Edit the font for that specific element. There are three settings which can be changed: the font family, the size, and the formatting (bold or italic).



6. Click **OK** to save the profile.

7. Restart the Console to apply the changes.

To load and use a saved font profile, open the **Font** tab in the **Preference** dialog, and simply select the font profile to use and click **OK**.

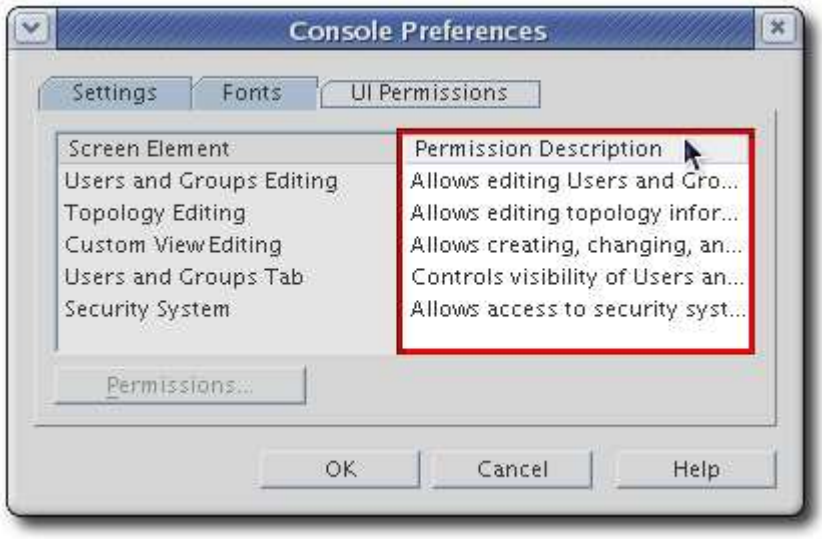


To delete a font profile, simply make sure that it is selected from the drop-down menu in the **Fonts** tab, and click the **Remove** button.

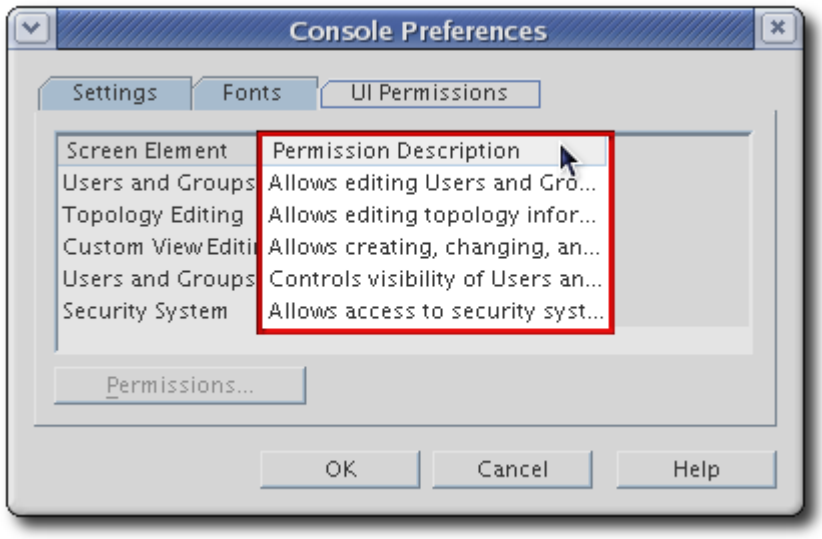
G.2.4.4. Reordering Table Columns

The columns in a table can be rearranged by dragging them into a new position.

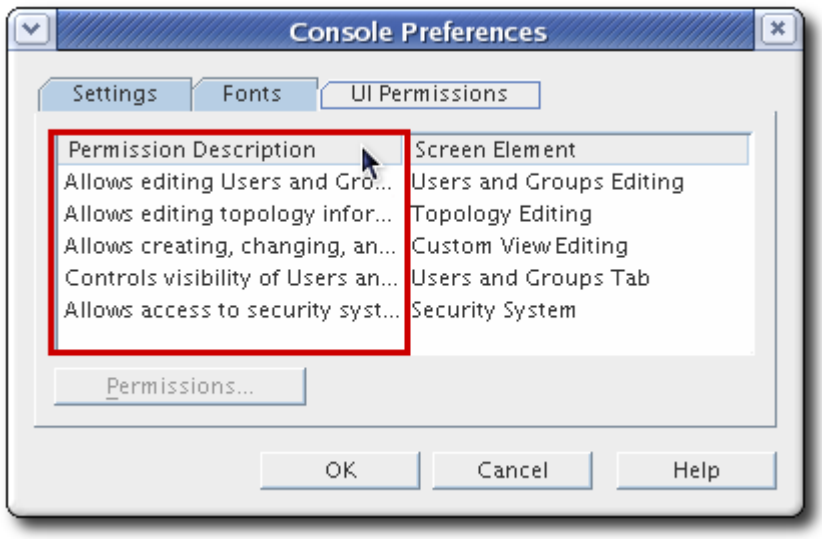
1. Click in the table heading.



2. Still holding down the left mouse button, drag the column to its new location. The other table columns will automatically shift down to their new positions.

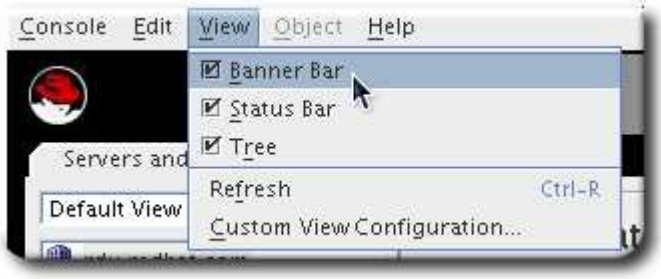


3. When you release the mouse button, the column snaps into its new position.

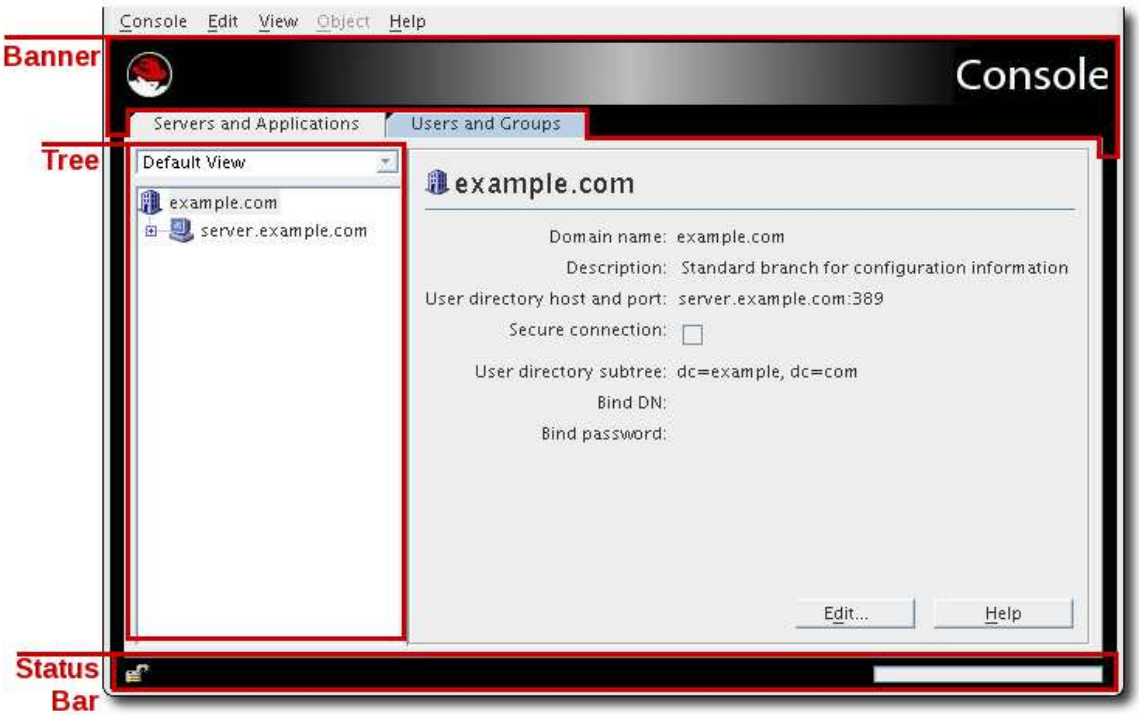


G.2.4.5. Customizing the Main Window

Different elements of the main Red Hat Management Console window can be displayed or hidden; this is set by check boxes in the **View** menu.



There are three parts of the Console which can be hidden: the navigation tree (the smaller panel on the left of the Console window); the decorative background and banner at the top of the Console window; and the status bar at the bottom of the Console.



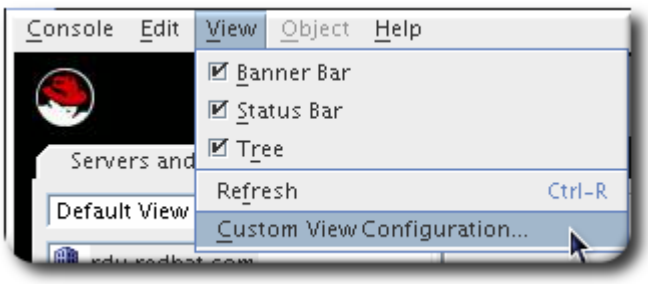
G.2.4.6. Working with Custom Views

The Console allows different *views* to be created to show different server and domain entries in the Red Hat Management Console window. Views show only a defined set of server entries; this makes it easier to maintain large numbers of instances or to have a quick way to perform specific tasks.

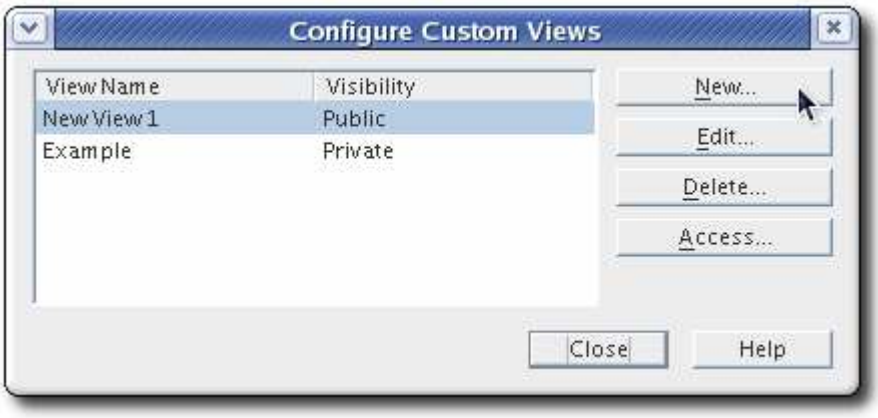
G.2.4.6.1. Creating Custom Views

Custom views show different, defined server instances. Views are either public or private. A public view is visible to any user, while a private view is visible only to the person who created it.

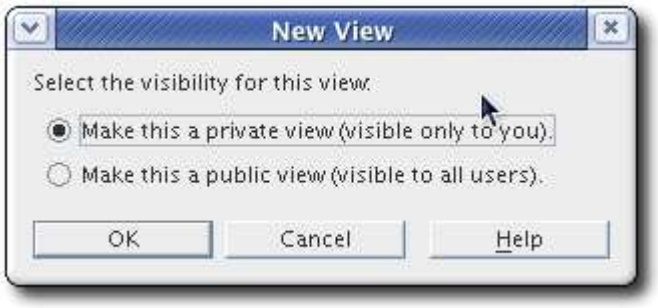
1. In the **View** menu, choose **Custom View Configuration**.



2. Click **New**.



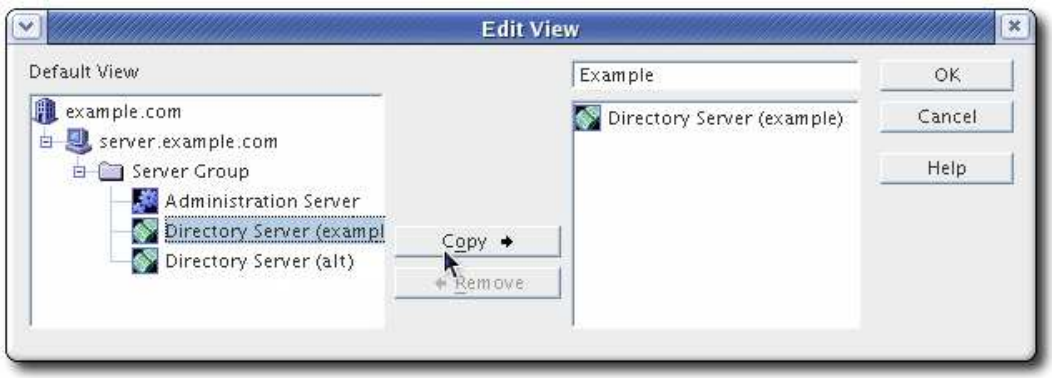
3. Choose whether the new view will be public or private, then click **OK**.



   - A public view is visible to all Console users by default, but access control instructions (ACIs) can be set to restrict access. For more information, see Section G.2.4.6.3, "Setting Access Permissions for a Public View".
   - A private view is only visible to the user who sets it, and ACIs cannot be set to change the access to it.

4. In the **Edit View** window, enter a descriptive name for this view.

5. Select a resource from the **Default View** navigation tree on the left. Click **Copy** to list it in the panel on the right and include it in the view.



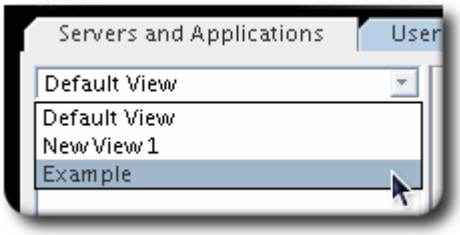
   To select a range of resources, click the **SHIFT** key and select the first and last entries; select multiple, separate resources by holding down the **Ctrl** key and selecting the entries.

To edit a custom view, select it from the list, click the **Edit** button, and make the changes to the name or resources.

To delete a custom view, select it from the list, and click the **Remove** button.

G.2.4.6.2. Switching to a Custom View

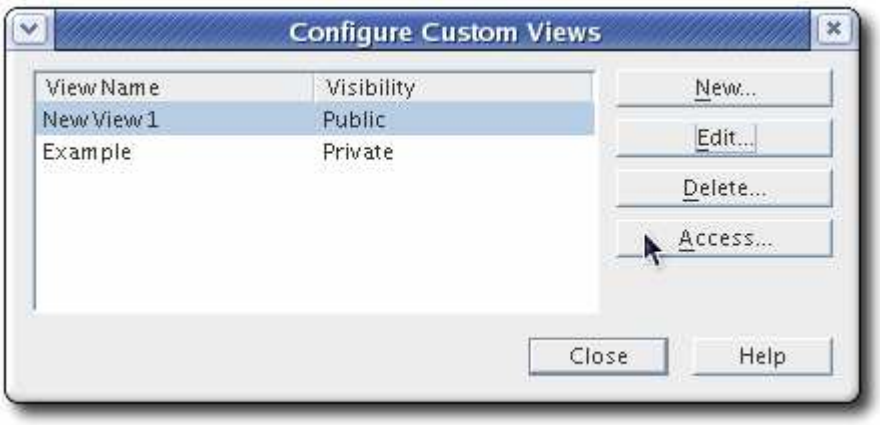
Choose the desired custom view from the drop-down list on the **Servers and Applications** tab.



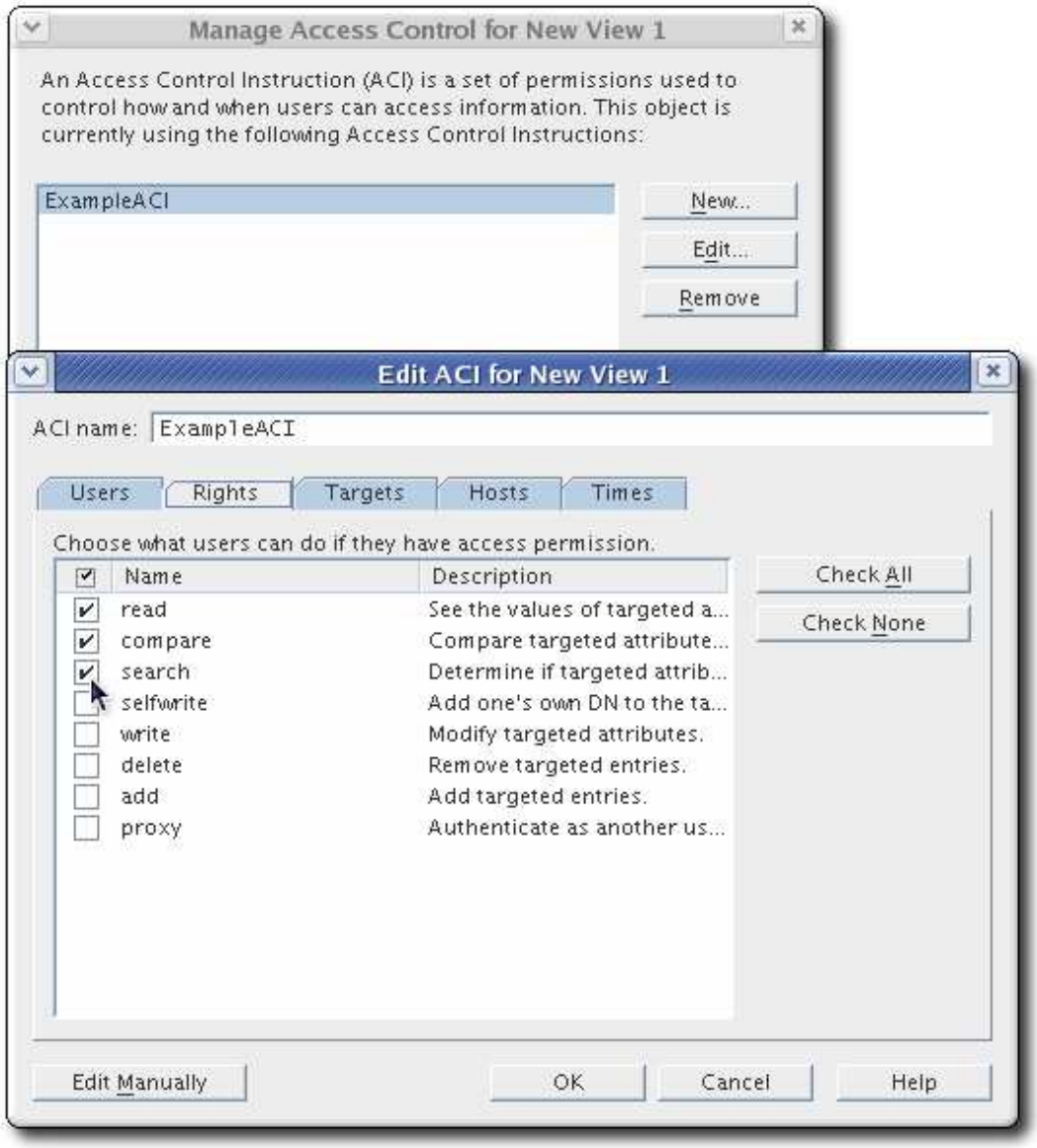
To return to the default view, choose **Default View** from the drop-down list.

G.2.4.6.3. Setting Access Permissions for a Public View

1. From the **View menu**, choose **Custom View Configuration**.
2. Choose a public **Custom View** from the list and click **Access**.



3. Set the access control instructions.



4. Click **OK** to save the ACI.

For more information on setting access permissions and creating access control instructions, see Section G.5, "Setting Access Controls".

G.3. MANAGING SERVER INSTANCES

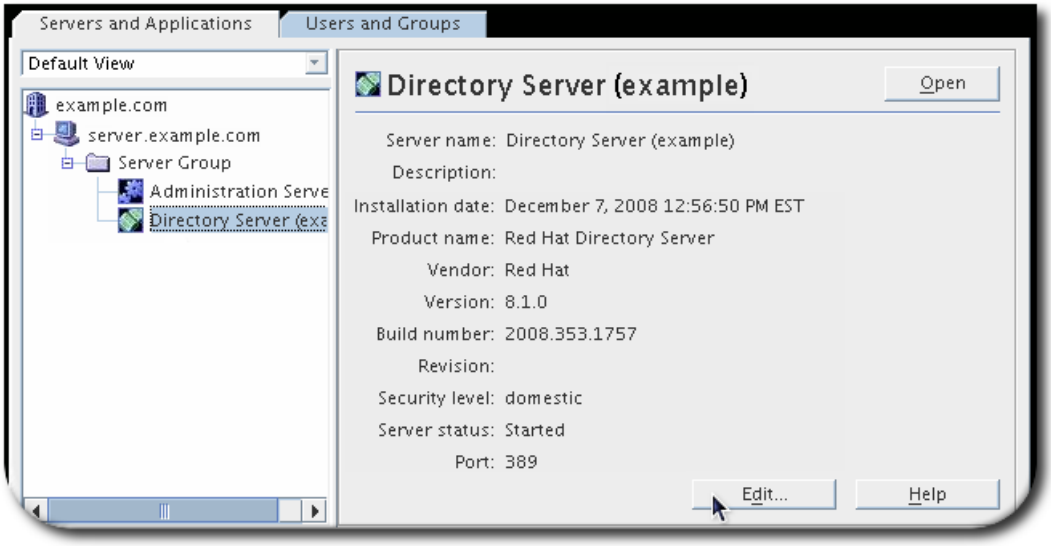
The server instances managed by the Red Hat Management Console are arranged in a hierarchy. At the top is the admin domain. Within the domain are hosts, representing different server machines. Each host has server groups, which identifies an inter-related group of Directory Servers using the same Admin Server instance. The individual Directory Server instances and a single Admin Server instance belong withing a server group. There can only be one Admin Server instance per server group.

These high level entries can be created and managed in the Red Hat Management Console.

G.3.1. Editing Domain, Host, Server Group, and Instance Information

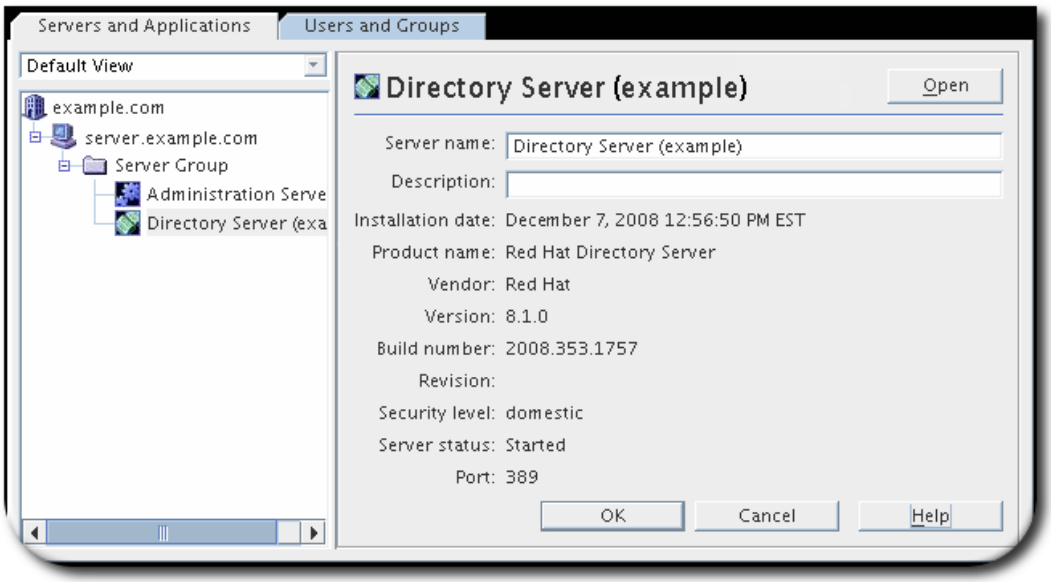
The Red Hat Console displays some information about every admin domain, host, group, and server instances. Most of this information – such as the installation date and build number – are not editable, but some information is.

1. In the **Servers and Applications** tab, select the entry to modify.



2. Click **Edit**.

3. Edit the instance's information. Every entry has the option to change its name and description. The host, which is the physical machine on which the instances are installed, also has the option of changing the location.



4. Click **OK**.

G.3.2. Creating and Removing Admin Domains

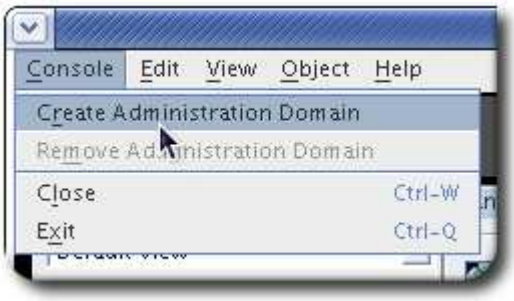
An admin domain is a container entry for server groups (and each server group contains Directory Server instances which are configured to work with the same Configuration Directory Server and the same Admin Server, which is also in the server group).

G.3.2.1. Creating and Editing an Admin Domain

To create a new admin domain:

1. In the top menu, click the **Console** menu item.

2. Select **Create New Administration Domain**.



3. Fill in the admin domain's information, including information for a new Directory Server instance.

4. Click **OK**.

To edit an admin domain, select the entry in the server window and click the **Edit** button.

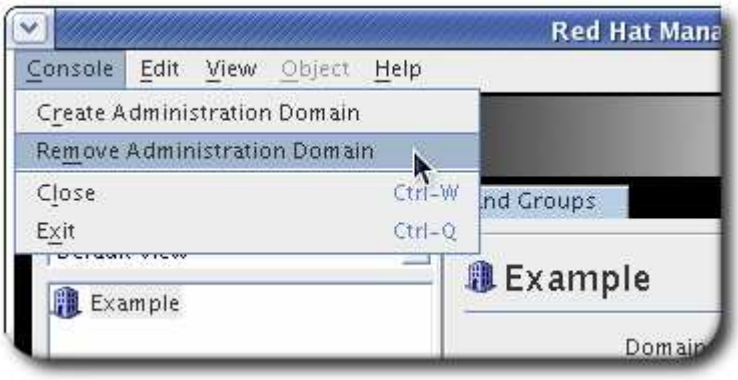
WARNING

The admin domain settings affect all servers within the domain. Making any changes to the admin domain settings means that all servers in the domain must be restarted.

G.3.2.2. Removing an Admin Domain

To remove an admin domain:

1. Highlight the admin domain to remove in the navigation tree.
2. In the top menu, click the **Console** menu item.
3. Select **Remove Administration Domain**.



4. Click **Yes**.

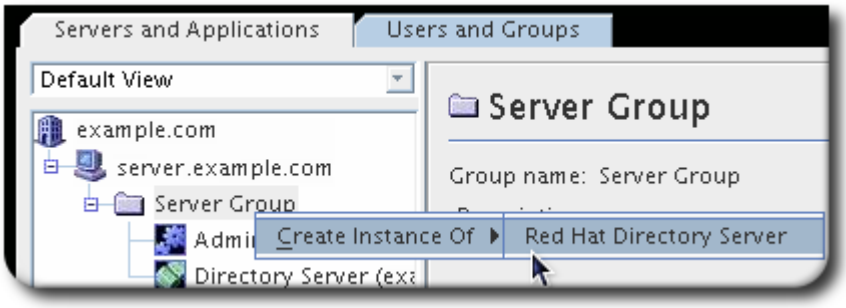
NOTE

Any server group and servers within the domain must be removed before the domain can be deleted.

G.3.3. Creating a New Directory Server Instance

After the default Red Hat Directory Server and Admin Server instances are installed and configured, additional Directory Server instances can be created using the same schema and configuration and in the same installation directory, **/etc/dirsrv**. Having multiple instances on the same host makes it easier to maintain divisions between directories while simplifying administering multiple directories.

1. In Red Hat Management Console, select the server group that will contain the new server instance.
2. Right-click on the server group entry, and select **Create Instance Of**, and then **Red Hat Directory Server**.



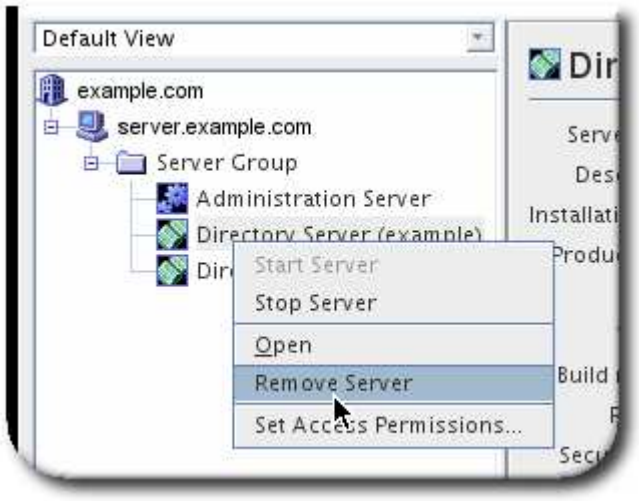
Alternatively, click **Object** in the top menu bar, and select **Create Instance Of**.

3. Fill in the information for the new instance of Directory Server, including the base DN, Directory Manager, and port.

4. Click **OK**.

G.3.4. Deleting a Directory Server Instance

1. In the Red Hat Management Console, select the instance to delete.
2. Right-click the server instance, and select **Remove Server**.



3. Click **Yes** to confirm the deletion.

G.4. MANAGING DIRECTORY SERVER USERS AND GROUPS

Users for both multiple Red Hat Directory Server instances and Admin Server can be created, edited, and searched for in the Red Hat Management Console. The main Console window can also be used to create organizational units and groups and to add entries to the new **ou**s and groups.

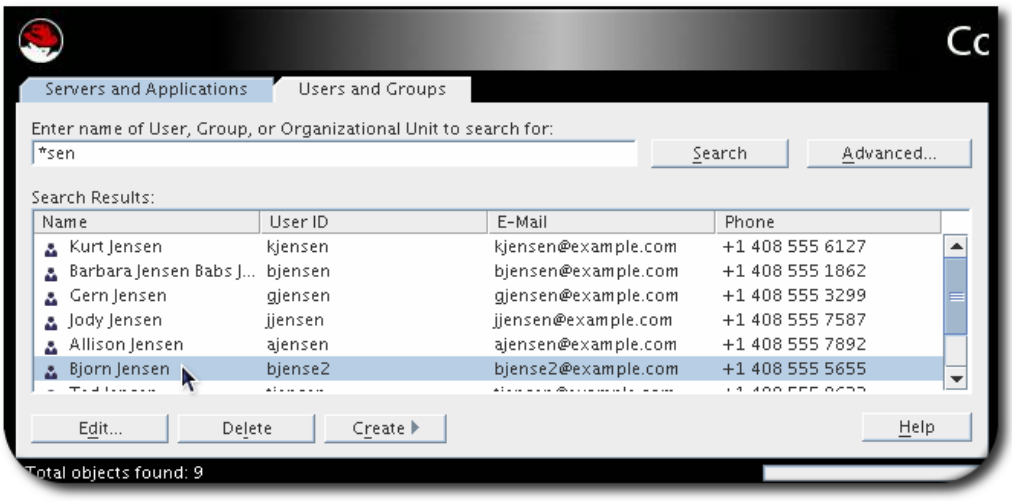
Section G.5, "Setting Access Controls" describes how to work with user and group information when setting access privileges and other security information.

G.4.1. Searching for Users and Groups

The **Users and Groups** searches for directory entries; by default, it looks in the default user directory configured for the Admin Server, but the directory can be changed to any Red Hat Directory Server instance.

To search the directory:

1. Click the **Users and Groups** tab.
2. Enter the search criteria, and click **Search**.
   - For a simple search, enter all or part of an entry name in the text box. To return all entries, leave the search field blank or enter an asterisk (*).

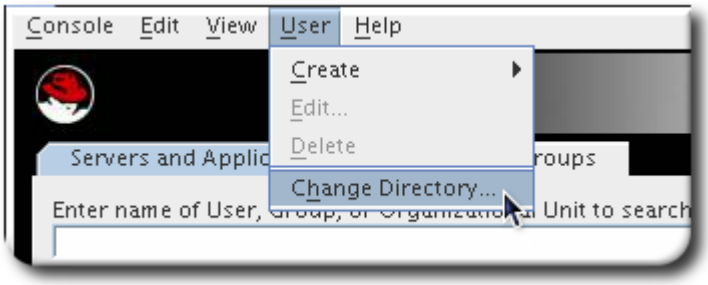


- For a more complex or focused search, click the **Advanced** button, and enter the attributes to search (such as ***cn***, ***givenname***, or ***ou***), the kind of search, and the search term. To add or remove search criteria, click the **More** and **Fewer** buttons.

3. Click **Search**. Results are displayed in the list box.

To change the search directory:

1. Click the **Users and Groups** tab.
2. In the top menu, select the **User** menu item, and choose **Change Directory**.



3. Fill in the user directory information.

- **User Directory Host**. The fully qualified host name for the Directory Server instance.
- **User Directory Port** and **Secure Connection**. The port number for the connection and whether this is an SSL (LDAPS).
- **User Directory Subtree**. The DN of the subtree to search in the directory; for example, **dc=example,dc=com** for the base DN or **ou=Marketing, dc=example,dc=com** for a subtree.
- **Bind DN** and **Bind Password**. The credentials to use to authenticate to the directory.

4. Click **OK**.

G.4.2. Creating Directory Entries

The Red Hat Management Console can be used to add, edit, and delete users, groups, and organization units in the **Users and Groups** tab. The different kinds of entries and options for creating entries is explained in more detail in the *Directory Server Administrator's Guide* .

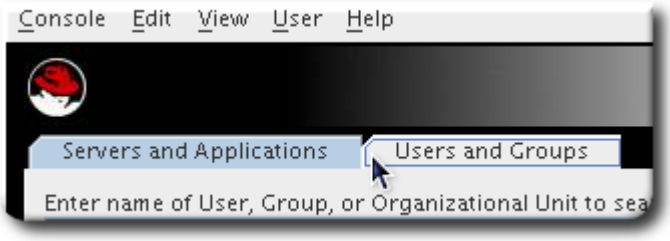
G.4.2.1. Directory and Administrative Users

NOTE

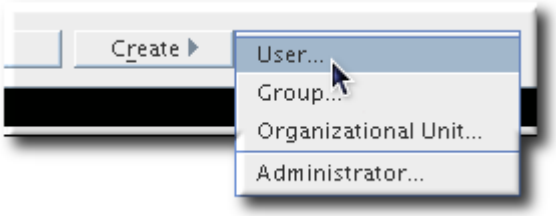
A user can be added to the Directory Server user database through the Console or a user can be added as an Admin Server administrator. The process is almost identical, with two exceptions:

- A Directory Server user is added by clicking the **Create** button, then the **Users** option, while an administrator is created by selecting the **Administrator** option.
- An administrator does not require selecting an organization unit, while the Directory Server user does, because the administrator is automatically added to **ou=Groups,ou=Topology,o=NetscapeRoot**.

1. Click the **Users and Groups** tab.

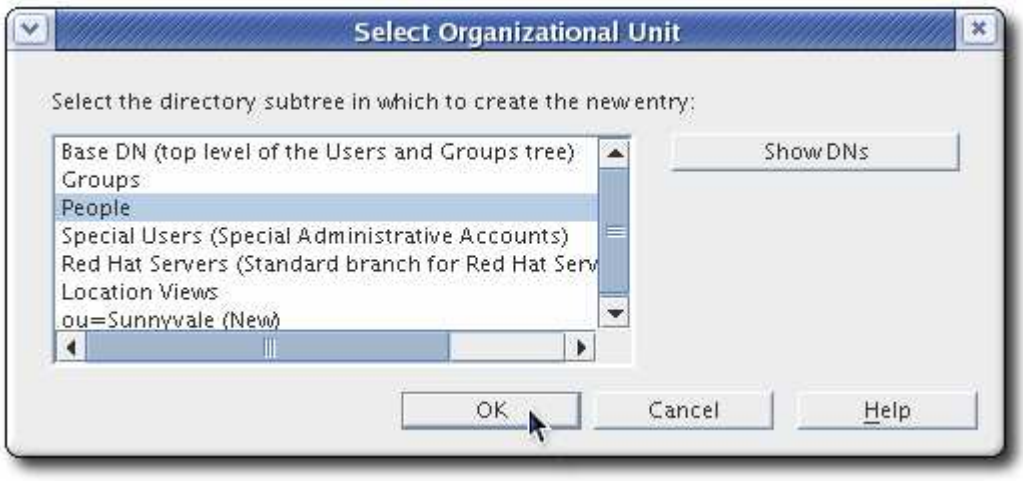


2. Click the **Create** button, and choose **User**.



Alternatively, open the **User** option in the top menu, and choose **Create > User**.

3. Select the are in the directory tree under which the entry is created.

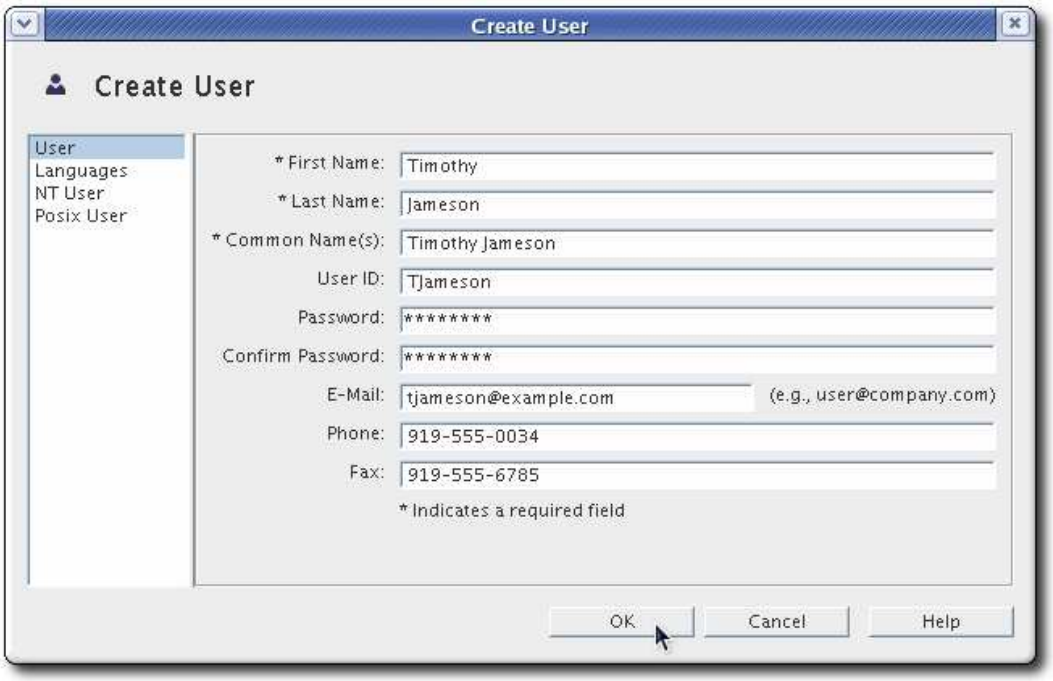


NOTE

When creating an administrator, there is no option to select the ***ou*** to which to add the user as there is with a regular Directory Server user. This is because the administrator is added to **ou=Groups,ou=Topology,o=NetscapeRoot**, with the admin users.

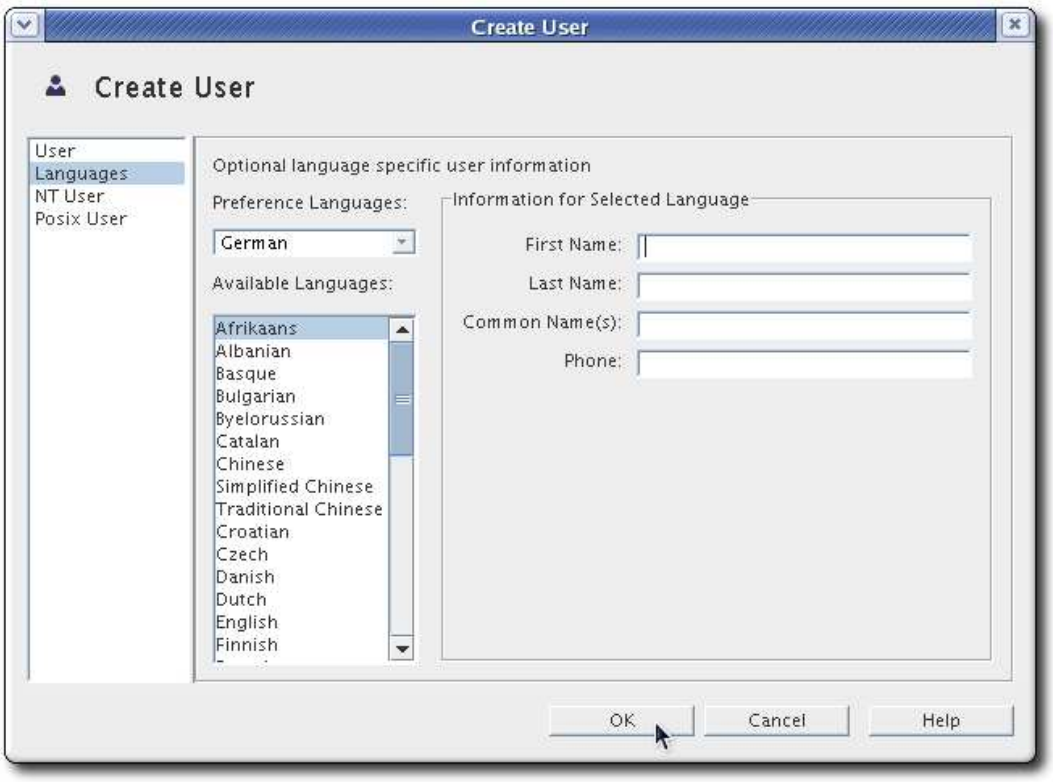
The entry can be added to an ***ou*** or a view, if views have been added to the directory.

4. In the **Create User** window, enter user information. The **Common Name** and **User ID** fields are automatically filled in with the combined values the **First Name** and **Last Name** fields. These first, last, and common name fields are required; a password is also required for the user to be able to log into the Directory Server and the Console, but is not a required attribute.



5. Optionally, click the **Languages** link on the left, select an alternate language and fill in internationalized values for common attributes.

   This option allows international users to select a language other than English and to represent their names in their preferred language. The pronunciation attribute allows for phonetic searching against the international name attributes.



6. Click **OK**.

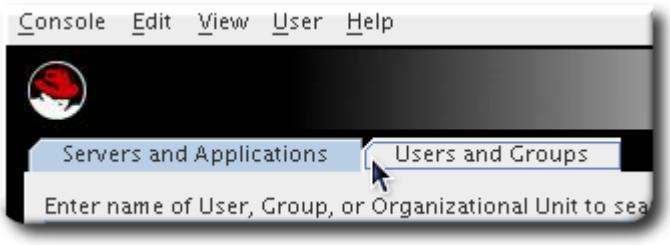
G.4.2.2. Groups

A group consists of users who share a common attribute or are part of a list. Red Hat Directory Server supports three types of groups: static, dynamic, and certificate. Each group differs by the way in which users, or *members*, are added to it:

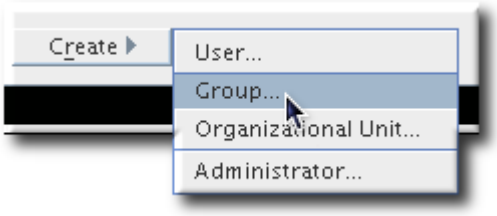
- A *static group* has members who are manually added to it, so it is *static* because the members do not change unless an administrator manually adds or removes users.
- A *dynamic group* automatically includes users based on one or more attributes in their entries; the attributes and values are determined using LDAP URLs. For example, a dynamic group can use an LDAP filter which searches for entries which contain the attributes and values **st=California** and **department=sales**. As entries are added to the directory with those two attributes, the users are automatically added as members to the dynamic group. If those attributes are removed from the entry, the entry is removed from the group.
- A *certificate group* includes all users who have a specific attribute-value pair in the subject name of the certificate. For example, the certificate group could be based on having the string **st=California,ou=Sales,ou=West** in the subject name. If a user logs onto a server using a certificate with those attributes in his certificate, the user is automatically added to the group and is granted all of the access privileges of that group.

To create a group:

1. Click the **Users and Groups** tab.



2. Click the **Create** button, and choose **Group**.



Alternatively, open the **User** option in the top menu, and choose **Create > Group**.

3. Select the are in the directory tree under which the entry is created.

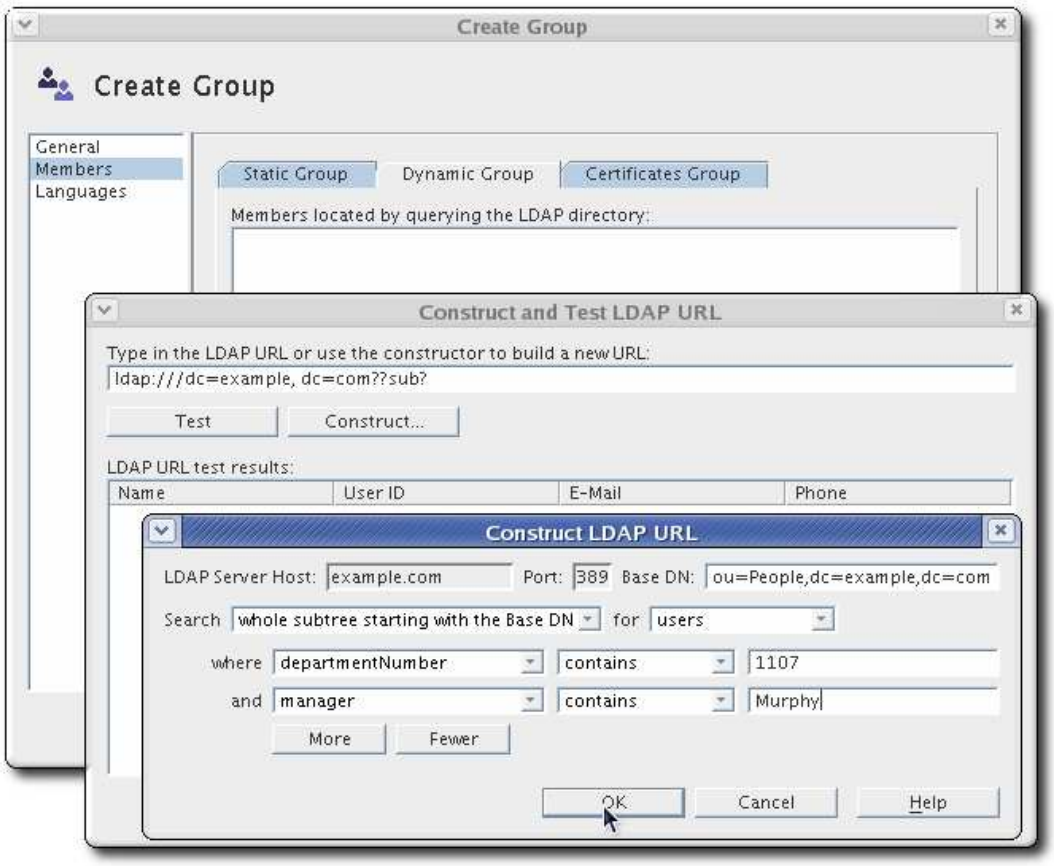
The subtree entry can be an ***ou*** or a view, if views have been added to the directory.

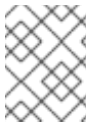
4. Enter the group's name and description.

It is possible to save the new group entry at this point, without adding members. Click **OK**.

5. Click the **Members** link to add members to the group, and click the tab of the type of group membership, **Static**, **Dynamic**, or **Certificate**.

6. Configure the members. For static groups, manually search for and add users; for dynamic groups, construct the LDAP URL to use to find entries; and for certificate groups, enter the values to search for in user certificate subject names.





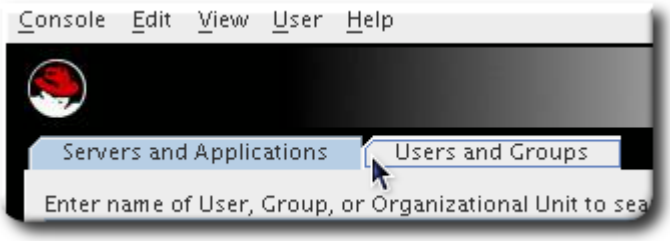
NOTE

The different kinds of groups and how to configure their members are explained in more detail in the *Directory Server Administrator's Guide* .

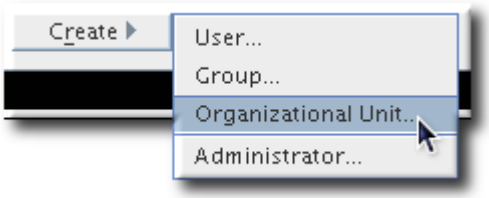
G.4.2.3. Organizational Units

An organizational unit can include a number of groups and users. An org unit usually represents a distinct, logical division in an organization, such as different departments or geographical locations. Each ***organizationalUnitName*** (***ou***) is a new subtree branch in the directory tree. This is reflected in the relative distinguished name of the ***ou***, such as **ou=People,dc=example,dc=com**, which becomes part of the distinguished names of its sub-entries.

1. Click the **Users and Groups** tab.



2. Click the **Create** button, and choose **Organizational Unit**.



Alternatively, open the **User** option in the top menu, and choose **Create > Organizational Unit**

3. Select the directory subtree under which to locate the new organizational unit.

4. Fill in the organizational unit information. The **Alias** offers an alternative name for the organizational unit that can be used instead of the full name.

5. Click **OK**.

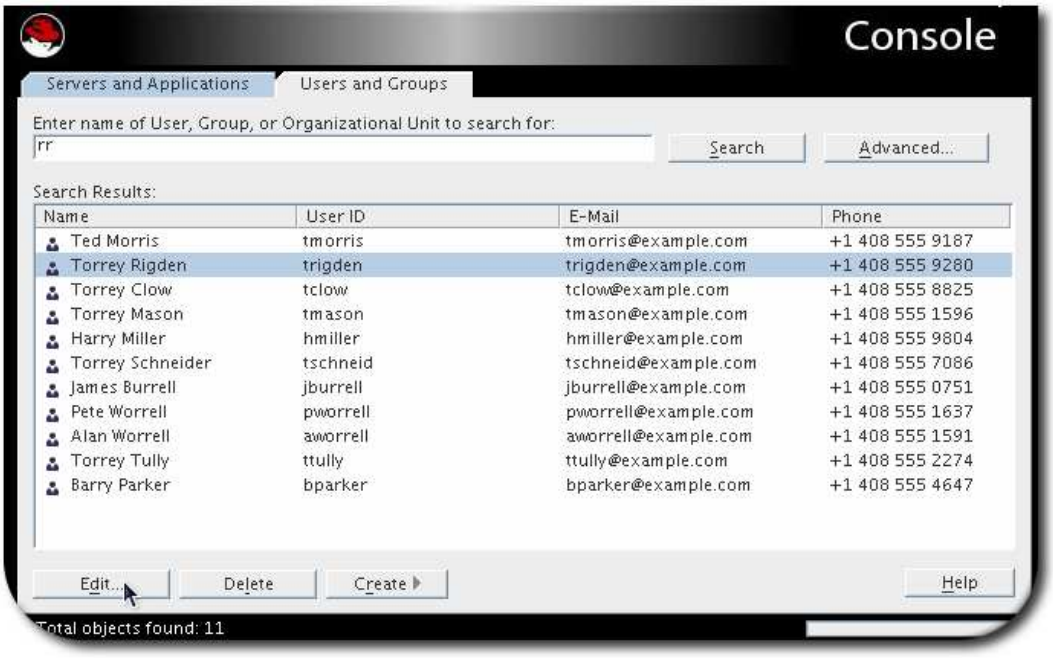
G.4.3. Modifying Directory Entries

G.4.3.1. Editing Entries

1. Search for the entry to edit.

   See Section G.4.1, "Searching for Users and Groups" for more information on searching for entries.

2. Select the entry, and click **Edit**.



3. Edit the entry information, and click **OK** to save the changes.

G.4.3.2. Allowing Sync Attributes for Entries

Red Hat Directory Server and Active Directory synchronization unify some Unix and Windows-specific directory attributes; to carry over a Directory Server entry to Active Directory, the entry must have **ntUser** attributes. (Likewise, Windows entries must have **posixAccount** attributes.)

Windows (NT) attributes must an be enabled on entries. By default, these attributes are added manually to individual entries. The user edit windows have links on the left for **NT User** to allow Directory Server entries to contain Windows-specific attributes for synchronization.

It is also possible to configure the server so that all new entries will automatically possess the **ntUser** object class; this is described in the Directory Server–Active Directory synchronization chapter of the *Directory Server Administrator's Guide*.

NOTE

Any Red Hat Directory Server entry must have the **ntUser** object class and required attributes added in order to be synchronized to Active Directory.

To enable synchronization:

1. Select or create a user, and click the **NT User** link.
2. Enable the NT account, and check how the entry will be synchronized (meaning, whether a new entry will be created and whether that entry should be deleted on Active Directory if it is delete on Directory Server).

3. Click **OK**.

G.4.3.3. Changing Administrator Entries

When the Admin Server is installed, two entries are created with administrator access in the Console. The main entry is the *Configuration Administrator*, who is authorized to access and modify the entire configuration directory (**o=NetscapeRoot**). The Configuration Administrator entry is stored in the **uid=***username*, **ou=Administrators,ou=TopologyManagement,o=NetscapeRoot** entry.

The Configuration Administrator's user name and password are automatically used to create the *Admin Server Administrator*, who can perform a limited number of tasks, such as starting, stopping, and restarting servers. The Admin Server Administrator is created so that a user can log into the Red Hat Management Console when the Directory Server is not running. The Admin Server Administrator does not have an LDAP entry; it exists in the Admin Server's configuration file, **/usr/share/dirsrv/properties/admpw**.

IMPORTANT

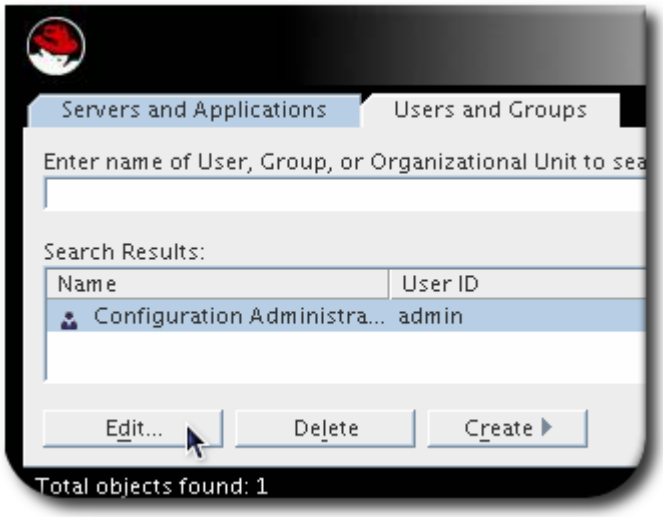
Even though they are created at the same time during installation, and are identical at that time, the Configuration Administrator and Admin Server Administrator are two separate entities. If the user name or password is changed for one, Red Hat Management Console does not automatically make the same changes for the other.

- Section G.4.3.3.1, "Changing the Configuration Administrator and Password"
- Section G.4.3.3.2, "Changing the Admin Password"
- Section G.4.3.3.3, "Adding Users to the Configuration Administrators Group"

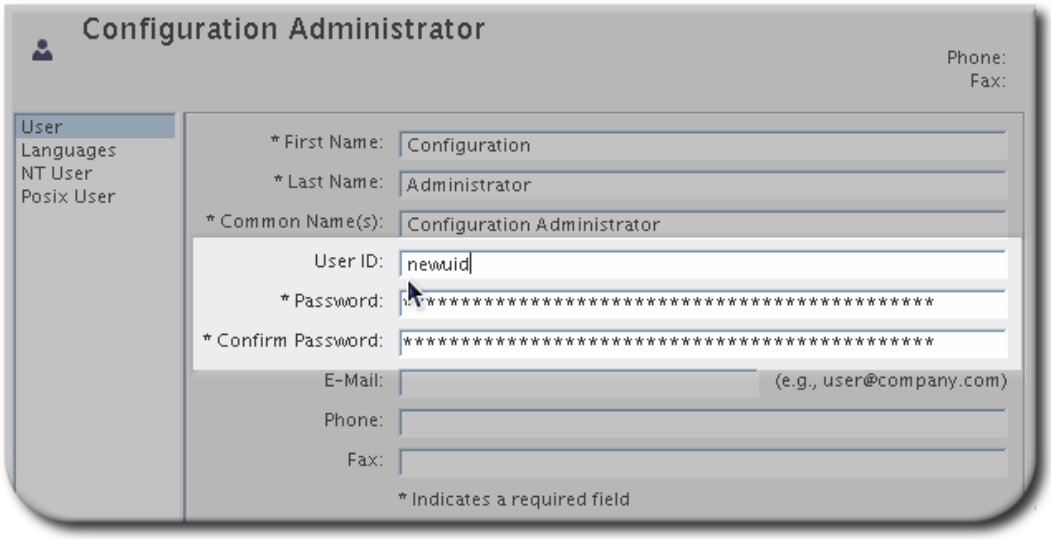
G.4.3.3.1. Changing the Configuration Administrator and Password

1. In the **Users and Groups**, click **Advanced**.
2. Search for the Configuration Administrator. Select the **Administrators** object, and enter the administrator's user name, **Configuration Administrator** by default.

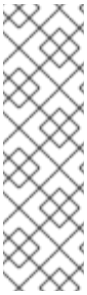
3. Select the Configuration Administrator from the list of search results, and then click **Edit**.



4. Change the administrator's ***uid*** and password. The ***uid*** is the naming attribute used to log into the Console and run commands.



5. Click **OK**.



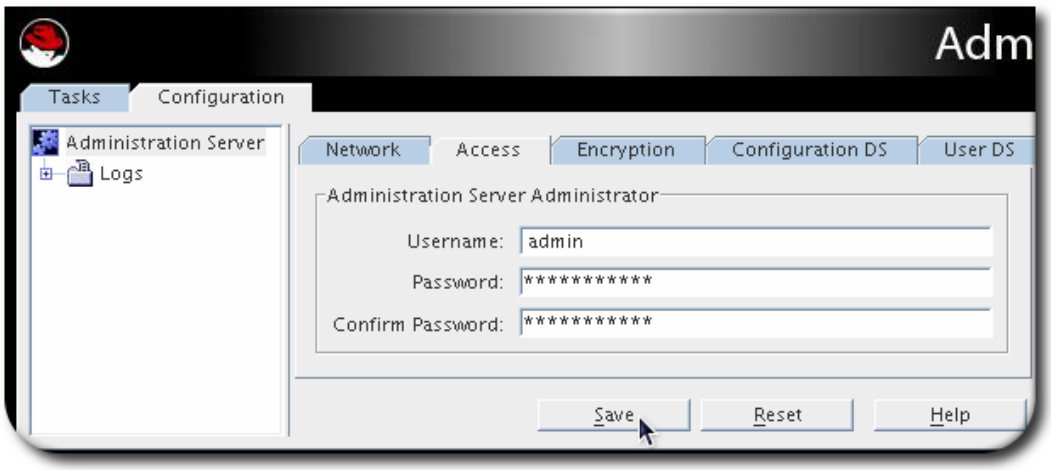
NOTE

If you are logged into the Console as the Configuration Administrator when you edited the Configuration Administrator entry, update the login information for the directory.

1. In the **Users and Groups** tab, click the **User** menu in the top menu and select **Change Directory**.
2. Update the **Bind DN** and **Bind Password** fields with the new information for the Configuration Administrator, and click **OK**.

G.4.3.3.2. Changing the Admin Password

1. Select the Admin Server in the **Servers and Applications** tab, and click **Open**.
2. Click the **Configuration** tab, and open the **Access** tab.
3. Set the new password.



WARNING

Do not change the admin user name.

4. Click **Save**.

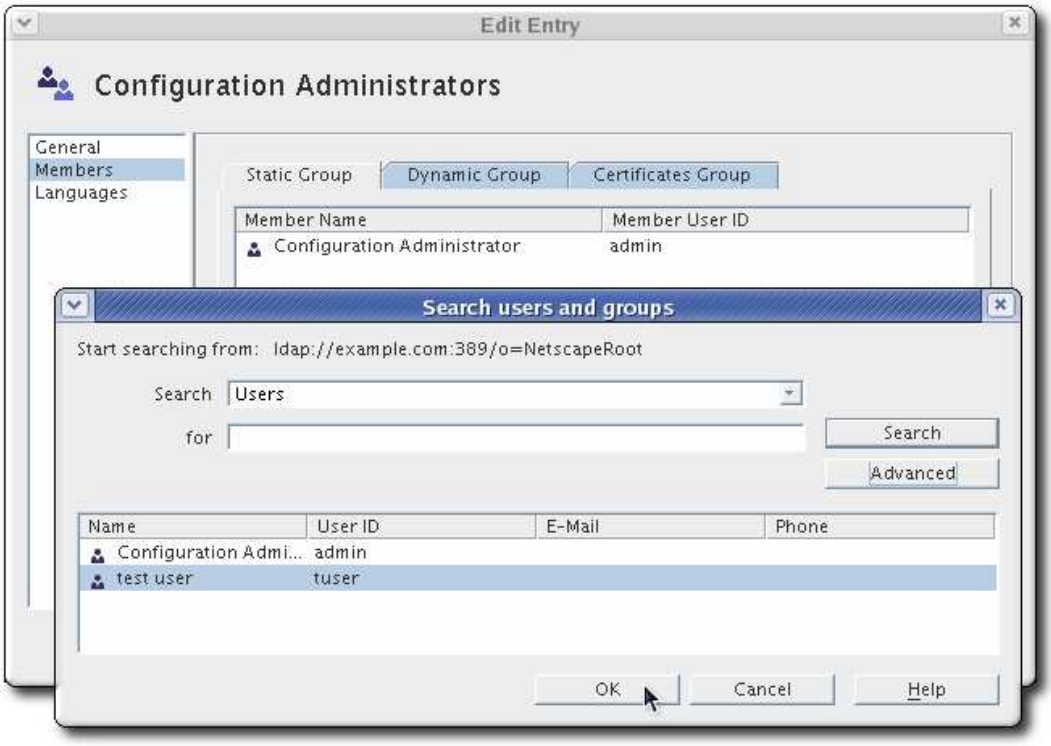
5. Restart the Admin Server.

```
service dirsrv-admin restart
```

G.4.3.3.3. Adding Users to the Configuration Administrators Group

1. In the **Users and Groups** tab, click the **User** menu in the top menu and select **Change Directory**.
2. Change to the **o=NetscapeRoot** subtree, which contains the configuration information and the Configuration Administrators group.

3. Search for the **Configuration Administrators** group, and click **Edit**.
4. Click the **Members** link in the left of the edit window.
5. Click **Add**, and search for the user to add to the group.



NOTE

Only users in the **o=NetscapeRoot** database can be added to the Configuration Administrators group. This means that the entry must be created as an administrator, not a regular user, when added through the Console. See Section G.4.2.1, "Directory and Administrative Users".

G.4.3.4. Removing an Entry from the Directory

1. Search for the entry to deleted.

   See Section G.4.1, "Searching for Users and Groups" for more information on searching for entries.

   NOTE

   All entries must be removed from under an organization unit before it can be deleted.

2. Select the entry in the results list, and click **Delete**. Click **OK** to confirm the deletion.

G.5. SETTING ACCESS CONTROLS

Access control instructions (ACIs) can be set in the Red Hat Management Console to set limits on what users can see and what operations they can perform on Red Hat Directory Server and Admin Server instances managed in the Console.

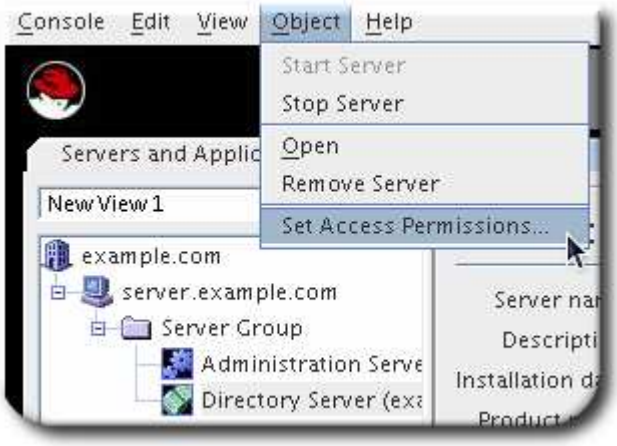
ACIs define what operations users can do with a specific instance of Red Hat Directory Server or Admin Server. ACIs set rules on areas of the subtree which can be accessed or modified, what operations are allowed, even what hosts can be used to access the server and what times of day access is allowed.

For Red Hat Management Console, access controls can be used to grant administrative privileges very easily to specific users and to set restrictions on different aspects of the main Console, such as searching the directory, adding and editing users and groups, and editing server or Console settings.

G.5.1. Granting Admin Privileges to Users for Directory Server and Admin Server

Users can be granted administrative privileges, the same as the **admin** user for the Admin Server and similar to the **cn=Directory Manager** user in Directory Server (though not exactly the same as the Directory Manager, which is a special user).

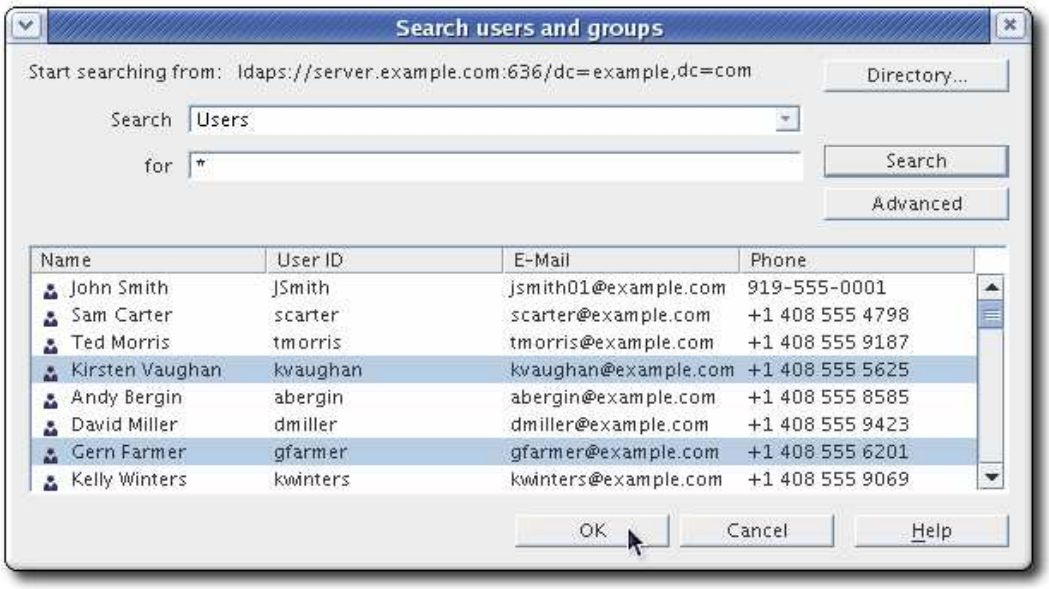
1. Highlight a server in the Console navigation tree.
2. Select the **Object** menu, and choose **Set Access Permissions**.



   Alternatively, right-click the entry, and choose **Set Access Permissions**.

3. Click **Add** to add a new user to the list of administrators for the server. The default users, **Directory Manager** for the Directory Server and **admin** for the Admin Server, are not listed in the **Set Permissions Dialog** box.

4. Search for the users to add as an administrators. In the results, highlight the selected users, and click **Add** to add them to the administrators list.



For more information on searching for users and groups, see Section G.4.1, "Searching for Users and Groups".

5. Click **OK** to add the names to the **Set Permissions Dialog** list, then click **OK** again to save the changes and close the dialog.

NOTE

Granting a user the right to administer a server does not automatically allow that user to give others the same right. To allow a user to grant administrative rights to other users, add that user to the Configuration Administrators group, as described in Section G.4.3.3.3, "Adding Users to the Configuration Administrators Group".

G.5.2. Setting Access Permissions on Console Elements

There are five elements defined in the Console for access control rules:

- User and Groups Tab (viewing)
- User and Groups Tab (editing)
- Topology Tab (editing)
- Custom View Tab (editing)
- Server Security (editing)

By default, each of these Console elements has five inherited ACIs:

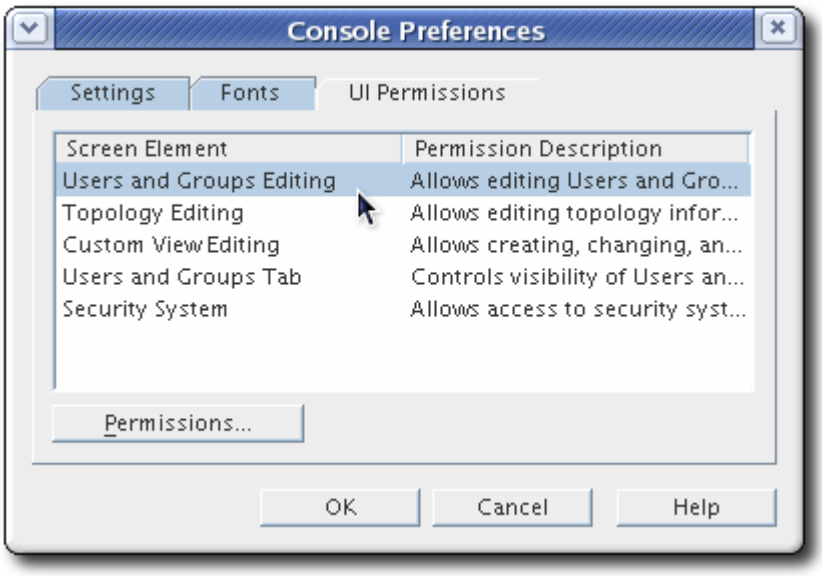
- Enabling anonymous access
- Default anonymous access
- Configuration administrator's modifications
- Enabling group expansions
- SIE (host) group permissions

These inherited ACIs cannot be edited, but new ACIs can be added for each Console element in addition to these defaults. Additional ACIs can limit anonymous access, for example, and change other permissions within the Red Hat Management Console, which, in turn, affects access to the Directory Server and Admin Server instances.

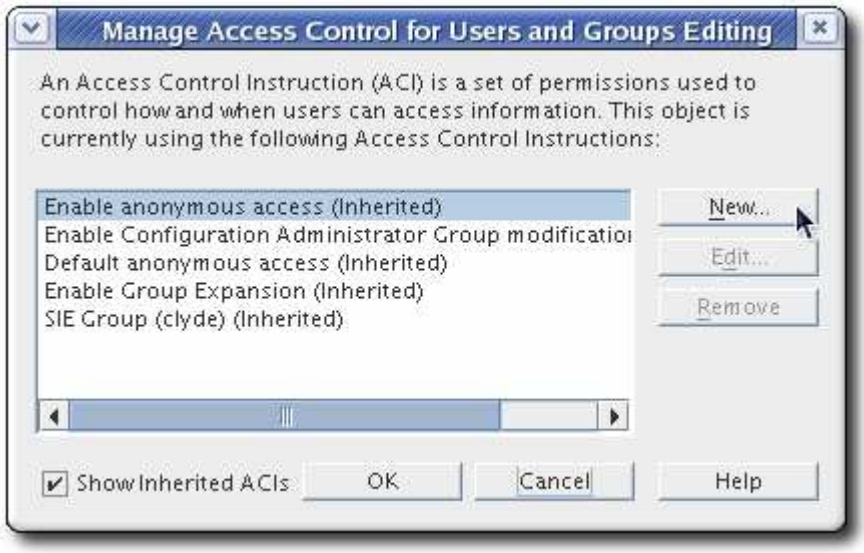
To create new ACIs:

1. In the top menu, select **Edit** and then **Preferences**.

2. Select the Console element from the list, and click the **Permissions** button.



3. In the **ACI Manager** window, click the **New** button.

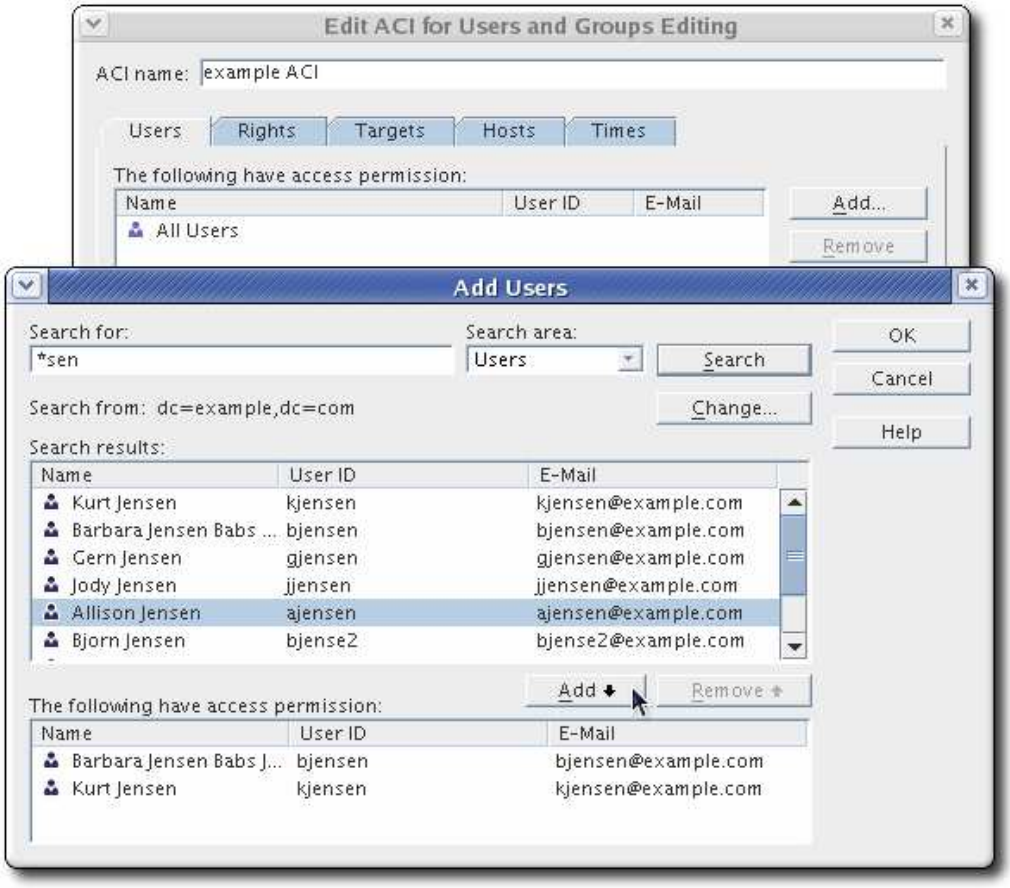


The five inherited ACIs are not displayed by default; to see them listed, click the **Show inherited ACIs** check box.

4. Configure the ACI by setting, at a minimum, the users to which it applies and the rights which are allowed. To configure the ACI in the wizard (visually):

    1. Enter a name for the ACI in the **ACI Name** field.

    2. In the **Users/Groups tab**, click the **Add** button to open the search window. Search for and add the users to which apply the ACI.



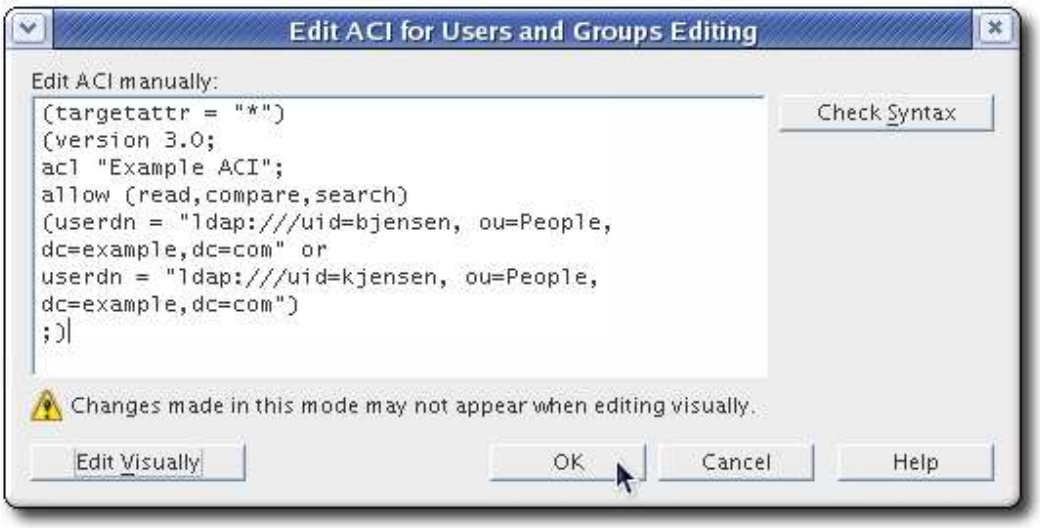
Select the users from the results list and click the **Add** button to include them. Click **OK** to save the list.

3. In the **Rights** tab, specify which operations are permitted as part of this ACI.

To hide a Console element entirely from the selected users, groups, and hosts, click **Check None** to block any access.

4. Optionally, set the target entry in the subtree, host names, or times of day where the ACI is in effect.

More complex ACIs may not be able to be edited visually; in those cases, click the **Edit Manually** button, and configure the ACI entry directly.



Use the **Check syntax** button to validate the ACI.

5. Click **OK** to save the ACI.

6. Restart Red Hat Management Console to apply the new ACI.

GLOSSARY

A

access control instruction

See ACI.

access control list

See ACL.

access rights

In the context of access control, specify the level of access granted or denied. Access rights are related to the type of operation that can be performed on the directory. The following rights can be granted or denied: read, write, add, delete, search, compare, selfwrite, proxy and all.

account inactivation

Disables a user account, group of accounts, or an entire domain so that all authentication attempts are automatically rejected.

ACI

An instruction that grants or denies permissions to entries in the directory.

See Also access control instruction.

ACL

The mechanism for controlling access to your directory.

See Also access control list.

All IDs Threshold

Replaced with the ID list scan limit in Directory Server version 7.1. A size limit which is globally applied to every index key managed by the server. When the size of an individual ID list reaches this limit, the server replaces that ID list with an All IDs token.

See Also ID list scan limit.

All IDs token

A mechanism which causes the server to assume that all directory entries match the index key. In effect, the All IDs token causes the server to behave as if no index was available for the search request.

anonymous access

When granted, allows anyone to access directory information without providing credentials, and regardless of the conditions of the bind.

approximate index

Allows for efficient approximate or "sounds-like" searches.

attribute

Holds descriptive information about an entry. Attributes have a label and a value. Each attribute also follows a standard syntax for the type of information that can be stored as the attribute value.

attribute list

A list of required and optional attributes for a given entry type or object class.

authenticating directory server

In pass-through authentication (PTA), the authenticating Directory Server is the Directory Server that contains the authentication credentials of the requesting client. The PTA-enabled host sends PTA requests it receives from clients to the host.

authentication

(1) Process of proving the identity of the client user to the Directory Server. Users must provide a bind DN and either the corresponding password or certificate in order to be granted access to the directory. Directory Server allows the user to perform functions or access files and directories based on the permissions granted to that user by the directory administrator.

(2) Allows a client to make sure they are connected to a secure server, preventing another computer from impersonating the server or attempting to appear secure when it is not.

authentication certificate

Digital file that is not transferable and not forgeable and is issued by a third party. Authentication certificates are sent from server to client or client to server in order to verify and authenticate the other party.

B

base distinguished name

See base DN.

base DN

Base distinguished name. A search operation is performed on the base DN, the DN of the entry and all entries below it in the directory tree.

bind distinguished name

See bind DN.

bind DN

Distinguished name used to authenticate to Directory Server when performing an operation.

bind rule

In the context of access control, the bind rule specifies the credentials and conditions that a particular user or client must satisfy in order to get access to directory information.

branch entry

An entry that represents the top of a subtree in the directory.

browser

Software, such as Mozilla Firefox, used to request and view World Wide Web material stored as HTML files. The browser uses the HTTP protocol to communicate with the host server.

browsing index

Speeds up the display of entries in the Directory Server Console. Browsing indexes can be created on any branch point in the directory tree to improve display performance.

See Also virtual list view index .

C

CA

See Certificate Authority.

cascading replication

In a cascading replication scenario, one server, often called the hub supplier, acts both as a consumer and a supplier for a particular replica. It holds a read-only replica and maintains a changelog. It receives updates from the supplier server that holds the master copy of the data and in turn supplies those updates to the consumer.

certificate

A collection of data that associates the public keys of a network user with their DN in the directory. The certificate is stored in the directory as user object attributes.

Certificate Authority

Company or organization that sells and issues authentication certificates. You may purchase an authentication certificate from a Certification Authority that you trust. Also known as a CA.

CGI

Common Gateway Interface. An interface for external programs to communicate with the HTTP server. Programs written to use CGI are called CGI programs or CGI scripts and can be written in many of the common programming languages. CGI programs handle forms or perform output parsing that is not done by the server itself.

chaining

A method for relaying requests to another server. Results for the request are collected, compiled, and then returned to the client.

changelog

A changelog is a record that describes the modifications that have occurred on a replica. The supplier server then replays these modifications on the replicas stored on replica servers or on other masters, in the case of multi-master replication.

character type

Distinguishes alphabetic characters from numeric or other characters and the mapping of upper-case to lower-case letters.

ciphertext

Encrypted information that cannot be read by anyone without the proper key to decrypt the information.

class definition

Specifies the information needed to create an instance of a particular object and determines how the object works in relation to other objects in the directory.

class of service

See CoS.

classic CoS

A classic CoS identifies the template entry by both its DN and the value of one of the target entry's attributes.

client

See LDAP client.

code page

An internal table used by a locale in the context of the internationalization plug-in that the operating system uses to relate keyboard keys to character font screen displays.

collation order

Provides language and cultural-specific information about how the characters of a given language are to be sorted. This information might include the sequence of letters in the alphabet or how to compare letters with accents to letters without accents.

consumer

Server containing replicated directory trees or subtrees from a supplier server.

consumer server

In the context of replication, a server that holds a replica that is copied from a different server is called a consumer for that replica.

CoS

A method for sharing attributes between entries in a way that is invisible to applications.

CoS definition entry

Identifies the type of CoS you are using. It is stored as an LDAP subentry below the branch it affects.

CoS template entry

Contains a list of the shared attribute values.

See Also template entry.

D

daemon

A background process on a Unix machine that is responsible for a particular system task. Daemon processes do not need human intervention to continue functioning.

DAP

Directory Access Protocol. The ISO X.500 standard protocol that provides client access to the directory.

data master

The server that is the master source of a particular piece of data.

database link

An implementation of chaining. The database link behaves like a database but has no persistent storage. Instead, it points to data stored remotely.

default index

One of a set of default indexes created per database instance. Default indexes can be modified, although care should be taken before removing them, as certain plug-ins may depend on them.

definition entry

See CoS definition entry.

Directory Access Protocol

See DAP.

Directory Manager

The privileged database administrator, comparable to the root user in UNIX. Access control does not apply to the Directory Manager.

directory service

A database application designed to manage descriptive, attribute-based information about people and resources within an organization.

directory tree

The logical representation of the information stored in the directory. It mirrors the tree model used by most filesystems, with the tree's root point appearing at the top of the hierarchy. Also known as DIT.

distinguished name

String representation of an entry's name and location in an LDAP directory.

DIT

See directory tree.

DM

See Directory Manager.

DN

See distinguished name.

DNS

Domain Name System. The system used by machines on a network to associate standard IP addresses (such as 198.93.93.10) with host names (such as **www.example.com**). Machines normally get the IP address for a host name from a DNS server, or they look it up in tables maintained on their systems.

DNS alias

A DNS alias is a host name that the DNS server knows points to a different hostspecifically a DNS CNAME record. Machines always have one real name, but they can have one or more aliases. For example, an alias such as **www.***yourdomain.domain* might point to a real machine called **realthing.***yourdomain.domain* where the server currently exists.

E

entry

A group of lines in the LDIF file that contains information about an object.

entry distribution

Method of distributing directory entries across more than one server in order to scale to support large numbers of entries.

entry ID list

Each index that the directory uses is composed of a table of index keys and matching entry ID lists. The entry ID list is used by the directory to build a list of candidate entries that may match the client application's search request.

equality index

Allows you to search efficiently for entries containing a specific attribute value.

F

file extension

The section of a filename after the period or dot (.) that typically defines the type of file (for example, .GIF and .HTML). In the filename **index.html** the file extension is **html**.

file type

The format of a given file. For example, graphics files are often saved in GIF format, while a text file is usually saved as ASCII text format. File types are usually identified by the file extension (for example, .GIF or .HTML).

filter

A constraint applied to a directory query that restricts the information returned.

filtered role

Allows you to assign entries to the role depending upon the attribute contained by each entry. You do this by specifying an LDAP filter. Entries that match the filter are said to possess the role.

G

general access

When granted, indicates that all authenticated users can access directory information.

GSS-API

Generic Security Services. The generic access protocol that is the native way for UNIX-based systems to access and authenticate Kerberos services; also supports session encryption.

H

host name

A name for a machine in the form machine.domain.dom, which is translated into an IP address. For example, **www.example.com** is the machine **www** in the subdomain **example** and **com** domain.

HTML

Hypertext Markup Language. The formatting language used for documents on the World Wide Web. HTML files are plain text files with formatting codes that tell browsers such as the Mozilla Firefox how to display text, position graphics, and form items and to display links to other pages.

HTTP

Hypertext Transfer Protocol. The method for exchanging information between HTTP servers and clients.

HTTPD

An abbreviation for the HTTP daemon or service, a program that serves information using the HTTP protocol. The daemon or service is often called an httpd.

HTTPS

A secure version of HTTP, implemented using the Secure Sockets Layer, SSL.

hub

In the context of replication, a server that holds a replica that is copied from a different server, and, in turn, replicates it to a third server.

See Also cascading replication.

I

ID list scan limit

A size limit which is globally applied to any indexed search operation. When the size of an individual ID list reaches this limit, the server replaces that ID list with an all IDs token.

index key

Each index that the directory uses is composed of a table of index keys and matching entry ID lists.

indirect CoS

An indirect CoS identifies the template entry using the value of one of the target entry's attributes.

international index

Speeds up searches for information in international directories.

International Standards Organization

See ISO.

IP address

Also Internet Protocol address. A set of numbers, separated by dots, that specifies the actual location of a machine on the Internet (for example, 198.93.93.10). Directory Server supports both IPv4 and IPv6 IP addresses.

ISO

International Standards Organization.

K

knowledge reference

Pointers to directory information stored in different databases.

L

LDAP

Lightweight Directory Access Protocol. Directory service protocol designed to run over TCP/IP and across multiple platforms.

LDAP client

Software used to request and view LDAP entries from an LDAP Directory Server.

See Also browser.

LDAP Data Interchange Format

See LDAP Data Interchange Format.

LDAP URL

Provides the means of locating Directory Servers using DNS and then completing the query via LDAP. A sample LDAP URL is **ldap://ldap.example.com**.

LDAPv3

Version 3 of the LDAP protocol, upon which Directory Server bases its schema format.

LDBM database

A high-performance, disk-based database consisting of a set of large files that contain all of the data assigned to it. The primary data store in Directory Server.

LDIF

LDAP Data Interchange Format. Format used to represent Directory Server entries in text form.

leaf entry

An entry under which there are no other entries. A leaf entry cannot be a branch point in a directory tree.

Lightweight Directory Access Protocol

See LDAP.

locale

Identifies the collation order, character type, monetary format and time / date format used to present data for users of a specific region, culture, and/or custom. This includes information on how data of a given language is interpreted, stored, or collated. The locale also indicates which code page should be used to represent a given language.

M

managed object

A standard value which the SNMP agent can access and send to the NMS. Each managed object is identified with an official name and a numeric identifier expressed in dot-notation.

managed role

Allows creation of an explicit enumerated list of members.

management information base

See MIB.

mapping tree

A data structure that associates the names of suffixes (subtrees) with databases.

master

See supplier.

master agent

See SNMP master agent.

matching rule

Provides guidelines for how the server compares strings during a search operation. In an international search, the matching rule tells the server what collation order and operator to use.

MD5

A message digest algorithm by RSA Data Security, Inc., which can be used to produce a short digest of data that is unique with high probability and is mathematically extremely hard to produce; a piece of data that will produce the same message digest.

MD5 signature

A message digest produced by the MD5 algorithm.

MIB

Management Information Base. All data, or any portion thereof, associated with the SNMP network. We can think of the MIB as a database which contains the definitions of all SNMP managed objects. The MIB has a tree-like hierarchy, where the top level contains the most general information about the network and lower levels deal with specific, separate network areas.

MIB namespace

Management Information Base namespace. The means for directory data to be named and referenced. Also called the directory tree.

monetary format

Specifies the monetary symbol used by specific region, whether the symbol goes before or after its value, and how monetary units are represented.

multi-master replication

An advanced replication scenario in which two servers each hold a copy of the same read-write replica. Each server maintains a changelog for the replica. Modifications made on one server are automatically replicated to the other server. In case of conflict, a time stamp is used to determine which server holds the most recent version.

multiplexor

The server containing the database link that communicates with the remote server.

N

n + 1 directory problem

The problem of managing multiple instances of the same information in different directories, resulting in increased hardware and personnel costs.

name collisions

Multiple entries with the same distinguished name.

nested role

Allows the creation of roles that contain other roles.

network management application

Network Management Station component that graphically displays information about SNMP managed devices, such as which device is up or down and which and how many error messages were received.

network management station

See NMS.

NIS

Network Information Service. A system of programs and data files that Unix machines use to collect, collate, and share specific information about machines, users, filesystems, and network parameters throughout a network of computers.

NMS

Powerful workstation with one or more network management applications installed. Also network management station.

ns-slapd

Red Hat's LDAP Directory Server daemon or service that is responsible for all actions of the Directory Server.

See Also slapd.

O

object class

Defines an entry type in the directory by defining which attributes are contained in the entry.

object identifier

A string, usually of decimal numbers, that uniquely identifies a schema element, such as an object class or an attribute, in an object-oriented system. Object identifiers are assigned by ANSI, IETF or similar organizations.

See Also OID.

OID

See object identifier.

operational attribute

Contains information used internally by the directory to keep track of modifications and subtree properties. Operational attributes are not returned in response to a search unless explicitly requested.

P

parent access

When granted, indicates that users have access to entries below their own in the directory tree if the bind DN is the parent of the targeted entry.

pass-through authentication

See PTA.

pass-through subtree

In pass-through authentication, the PTA directory server will pass through bind requests to the authenticating directory server from all clients whose DN is contained in this subtree.

password file

A file on Unix machines that stores Unix user login names, passwords, and user ID numbers. It is also known as **/etc/passwd** because of where it is kept.

password policy

A set of rules that governs how passwords are used in a given directory.

PDU

Encoded messages which form the basis of data exchanges between SNMP devices. Also protocol data unit.

permission

In the context of access control, permission states whether access to the directory information is granted or denied and the level of access that is granted or denied.

See Also access rights.

pointer CoS

A pointer CoS identifies the template entry using the template DN only.

presence index

Allows searches for entries that contain a specific indexed attribute.

protocol

A set of rules that describes how devices on a network exchange information.

protocol data unit

See PDU.

proxy authentication

A special form of authentication where the user requesting access to the directory does not bind with its own DN but with a proxy DN.

proxy DN

Used with proxied authorization. The proxy DN is the DN of an entry that has access permissions to the target on which the client-application is attempting to perform an operation.

PTA

Mechanism by which one Directory Server consults another to check bind credentials. Also pass-through authentication.

PTA directory server

In pass-through authentication (PTA), the PTA Directory Server is the server that sends (passes through) bind requests it receives to the authenticating directory server.

PTA LDAP URL

In pass-through authentication, the URL that defines the authenticating directory server, pass-through subtree(s), and optional parameters.

R

RAM

Random access memory. The physical semiconductor-based memory in a computer. Information stored in RAM is lost when the computer is shut down.

rc.local

A file on Unix machines that describes programs that are run when the machine starts. It is also called **/etc/rc.local** because of its location.

RDN

The name of the actual entry itself, before the entry's ancestors have been appended to the string to form the full distinguished name. Also relative distinguished name.

read-only replica

A replica that refers all update operations to read-write replicas. A server can hold any number of read-only replicas.

read-write replica

A replica that contains a master copy of directory information and can be updated. A server can hold any number of read-write replicas.

referential integrity

Mechanism that ensures that relationships between related entries are maintained within the directory.

referral

(1) When a server receives a search or update request from an LDAP client that it cannot process, it usually sends back to the client a pointer to the LDAP sever that can process the request.

(2) In the context of replication, when a read-only replica receives an update request, it forwards it to the server that holds the corresponding read-write replica. This forwarding process is called a referral.

relative distinguished name

See RDN.

replica

A database that participates in replication.

replica-initiated replication

Replication configuration where replica servers, either hub or consumer servers, pull directory data from supplier servers. This method is available only for legacy replication.

replication

Act of copying directory trees or subtrees from supplier servers to replica servers.

replication agreement

Set of configuration parameters that are stored on the supplier server and identify the databases to replicate, the replica servers to which the data is pushed, the times during which replication can occur, the DN and credentials used by the supplier to bind to the consumer, and how the connection is secured.

RFC

Request for Comments. Procedures or standards documents submitted to the Internet community. People can send comments on the technologies before they become accepted standards.

role

An entry grouping mechanism. Each role has *members*, which are the entries that possess the role.

role-based attributes

Attributes that appear on an entry because it possesses a particular role within an associated CoS template.

root

The most privileged user available on Unix machines. The root user has complete access privileges to all files on the machine.

root suffix

The parent of one or more sub suffixes. A directory tree can contain more than one root suffix.

S

SASL

An authentication framework for clients as they attempt to bind to a directory. Also Simple Authentication and Security Layer .

schema

Definitions describing what types of information can be stored as entries in the directory. When information that does not match the schema is stored in the directory, clients attempting to access the directory may be unable to display the proper results.

schema checking

Ensures that entries added or modified in the directory conform to the defined schema. Schema checking is on by default, and users will receive an error if they try to save an entry that does not conform to the schema.

Secure Sockets Layer

See SSL.

self access

When granted, indicates that users have access to their own entries if the bind DN matches the targeted entry.

Server Console

Java-based application that allows you to perform administrative management of your Directory Server from a GUI.

server daemon

The server daemon is a process that, once running, listens for and accepts requests from clients.

Server Selector

Interface that allows you select and configure servers using a browser.

server service

A process on Windows that, once running, listens for and accepts requests from clients. It is the SMB server on Windows NT.

service

A background process on a Windows machine that is responsible for a particular system task. Service processes do not need human intervention to continue functioning.

SIE

Server Instance Entry. The ID assigned to an instance of Directory Server during installation.

Simple Authentication and Security Layer

See SASL.

Simple Network Management Protocol

See SNMP.

single-master replication

The most basic replication scenario in which multiple servers, up to four, each hold a copy of the same read-write replicas to replica servers. In a single-master replication scenario, the supplier server maintains a changelog.

SIR

See supplier-initiated replication.

slapd

LDAP Directory Server daemon or service that is responsible for most functions of a directory except replication.

See Also ns-slapd.

SNMP

Used to monitor and manage application processes running on the servers by exchanging data about network activity. Also Simple Network Management Protocol.

SNMP master agent

Software that exchanges information between the various subagents and the NMS.

SNMP subagent

Software that gathers information about the managed device and passes the information to the master agent. Also called a subagent.

SSL

A software library establishing a secure connection between two parties (client and server) used to implement HTTPS, the secure version of HTTP. Also called Secure Sockets Layer.

standard index

index maintained by default.

sub suffix

A branch underneath a root suffix.

subagent

See SNMP subagent.

substring index

Allows for efficient searching against substrings within entries. Substring indexes are limited to a minimum of two characters for each entry.

suffix

The name of the entry at the top of the directory tree, below which data is stored. Multiple suffixes are possible within the same directory. Each database only has one suffix.

superuser

The most privileged user available on Unix machines. The superuser has complete access privileges to all files on the machine. Also called root.

supplier

Server containing the master copy of directory trees or subtrees that are replicated to replica servers.

supplier server

In the context of replication, a server that holds a replica that is copied to a different server is called a supplier for that replica.

supplier-initiated replication

Replication configuration where supplier servers replicate directory data to any replica servers.

symmetric encryption

Encryption that uses the same key for both encrypting and decrypting. DES is an example of a symmetric encryption algorithm.

system index

Cannot be deleted or modified as it is essential to Directory Server operations.

T

target

In the context of access control, the target identifies the directory information to which a particular ACI applies.

target entry

The entries within the scope of a CoS.

TCP/IP

Transmission Control Protocol/Internet Protocol. The main network protocol for the Internet and for enterprise (company) networks.

template entry

See CoS template entry.

time/date format

Indicates the customary formatting for times and dates in a specific region.

TLS

The new standard for secure socket layers; a public key based protocol. Also Transport Layer Security.

topology

The way a directory tree is divided among physical servers and how these servers link with one another.

Transport Layer Security

See TLS.

U

uid

A unique number associated with each user on a Unix system.

URL

Uniform Resource Locater. The addressing system used by the server and the client to request documents. It is often called a location. The format of a URL is *protocol://machine:port/document*. The port number is necessary only on selected servers, and it is often assigned by the server, freeing the user of having to place it in the URL.

V

virtual list view index

Speeds up the display of entries in the Directory Server Console. Virtual list view indexes can be created on any branch point in the directory tree to improve display performance.

See Also browsing index.

X

X.500 standard

The set of ISO/ITU-T documents outlining the recommended information model, object classes and attributes used by directory server implementation.

INDEX

A

B

C

D

E

G

H

I

M

N

O

P

S

T

U

V

W

APPENDIX H. REVISION HISTORY

Note that revision numbers relate to the edition of this manual, not to version numbers of Red Hat Directory Server.

Revision 9.1-13 **Jun 26, 2017** **Marc Muehlfeld**
Added a statement that this documentation is deprecated and no longer maintained.

Revision 9.1-12 **May 29, 2017** **Marc Muehlfeld**
Updated section "Setting Access Control for VLV Information".

Revision 9.1-11 **Mar 16, 2017** **Marc Muehlfeld**
Added section "The Replication Keep-alive Entry".

Revision 9.1-10 **Feb 24, 2017** **Marc Muehlfeld**

Added "Fine Grained ID List Size" and "Trimming the Replication Changelog" sections. Other minor fixes.

Revision 9.1-9 **Dec 15, 2016** **Marc Muehlfeld**
Updated Replacing Log Files with a Named Pipe section. Other minor fixes.

Revision 9.1-8 **Nov 11, 2016** **Marc Muehlfeld**
Updated supported JRE version. Added ACI for o=NetscapeRoot suffix for multi-master replication.

Revision 9.1-7 **Jun 21, 2016** **Petr Bokoč**
Added information to avoid using owner nobody:nobody. Updated Schema Replication section. Other minor fixes.

Revision 9.1-6 **June 30, 2013** **Ella Deon Lackey**
Added DN cache information and disk monitoring.

Revision 9.1-2 **March 8, 2013** **Ella Deon Lackey**
Updated example for setting root DN ACI. Other minor updates.

Revision 9.1-1 **February 21, 2013** **Ella Deon Lackey**
Updates for Red Hat Enterprise Linux 6.4.

Revision 9.0-3 **July 2, 2012** **Ella Deon Lackey**
Added Automembership Plug-in information. Added logconv options and logconv usage examples.

Revision 9.0-1 **January 30, 2012** **Ella Deon Lackey**
Added information for the Account Policy Plug-in.

Revision 9.0-0 **December 6, 2011** **Ella Deon Lackey**
Initial version for Red Hat Directory Server version 9.0.