CHAPTER 12. MANAGING THE DIRECTORY SCHEMA

Red Hat Directory Server comes with a standard schema that includes hundreds of object classes and attributes. While the standard object classes and attributes should meet most deployments' requirements, it can be necessary to extend the schema for specific directory data. Extending the schema is done by creating new object classes and attributes.

The *Red Hat Directory Server 10 Configuration, Command, and File Reference* is a reference for most the standard Directory Server attributes and object classes, with information on allowed and required attributes, which object classes take which attribute, and OID and value information. This is a good resource for identifying useful schema elements for a directory and determining what custom schema needs to be created.

12.1. OVERVIEW OF SCHEMA

The directory schema is a set of rules that defines how data can be stored in the directory. Directory information is stored discrete entries, and each entry is comprised of a set of attributes and their values. The kind of identity being described in the entry is defined in the entry's object classes. An object class specifies the kind of object the entry describes through the defined set of attributes for the object class.

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, including people, groups, locations, organizations and divisions, and equipment. The identity is described in a directory entries with attributes and their values, pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. Other aspects of the Directory Server configuration, including matching rules and LDAP controls, are also defined in the schema. All of these together are *schema elements*.

Every schema element is identified by a unique, dot-separated number. This is called the *object identifier* or *OID*.

12.1.1. Default Schema Files

The schema for Directory Server is defined in several different schema files (LDIF files which define schema elements). The Directory Server schema files are located in the **/usr/share/dirsrv/schema/** directory. The files in this directory are used as templates for new Directory Server instances. Adding a new schema into this directory will make it available to any new instances.

The attributes used by the Directory Server to perform operations and manage entries is described with other configuration settings in the *Red Hat Directory Server 10 Configuration, Command, and File Reference*.

12.1.2. Object Classes

In LDAP, an object class defines the set of attributes that can be used to define an entry. The LDAP standard provides object classes for many common types of entries, such as people (**person** and **inetOrgPerson**), groups (**groupOfNames**), locations (**locality**), organizations and divisions (**organization** and **organizationalUnit**), and equipment (**device**).

In a schema file, an object class is identified by the ***objectclasses*** line, then followed by its OID, name, a description, its direct superior object class (an object class which is required to be used in conjunction with the object class and which shares its attributes with this object class), and the list of required (**MUST**) and allowed (**MAY**) attributes.

This is shown in Example 12.1, “person Object Class Schema Entry” .

Example 12.1. person Object Class Schema Entry

```
objectClasses: ( 2.5.6.6 NAME 'person' DESC 'Standard LDAP objectclass' SUP top MUST ( sn $
cn ) MAY ( description $ seeAlso $ telephoneNumber $ userPassword ) X-ORIGIN 'RFC 4519' )
```

Every object class defines a number of required attributes (**MUST** keyword in the schema) and of allowed attributes (**MAY** keyword in the schema). Required attributes must be present in entries using the specified object class, while allowed attributes are permissible and available for the entry to use, but are not required for the entry to be valid.

As in Example 12.1, “person Object Class Schema Entry” , the **person** object class requires the ***cn***, ***sn***, and ***objectClass*** attributes and allows the ***description***, ***seeAlso***, ***telephoneNumber***, and ***userPassword*** attributes.

An object class can inherit attributes from another class, in addition to its own required and allowed attributes. The second object class is the *superior* or *parent* object class of the first.

For example, a user's entry has to have the **inetOrgPerson** object class. In that case, the entry must also include the superior object class for **inetOrgPerson**, **organizationalPerson**, and the superior object class for **organizationalPerson**, which is **person**:

```
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
```

12.1.3. Attributes

Directory entries are composed of attributes and their values. These pairs are called *attribute-value assertions* or AVAs. Any piece of information in the directory is associated with a descriptive attribute. For instance, the ***cn*** attribute is used to store a person's full name, such as **cn: John Smith**.

Additional attributes can supply additional information about John Smith:

```
givenname: John
surname: Smith
mail: jsmith@example.com
```

In a schema file, an attribute is described by:

- OID
- name
- syntax matching rule (optional)
- substring matching rules (optional)
- ordering rule (optional)
- description (optional)

- syntax
- single-valued or multi-valued attribute
- details about where the attribute is defined

This is shown in Example 12.2, "***uid*** Attribute Schema Entry" .

Example 12.2. *uid* Attribute Schema Entry

```
( 0.9.2342.19200300.100.1.1 NAME ( 'uid' 'userid' ) EQUALITY caseIgnoreMatch SUBSTR
caseIgnoreSubstringsMatch SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'RFC 4519' )
```

12.1.4. Extending the Schema

New, custom attributes and object classes can be added to a Directory Server instance to extend the schema, and there are several ways to add schema elements. Using the Directory Server Console or LDAP tools adds schema elements to the default custom schema file for an instance, **99user.ldif**. It is also possible to create a new, separate schema file and include it with the default schema files.

Adding new schema elements requires three things:

1. Planning and defining OIDs for the new schema. Schema elements are recognized by the server by their OID, so it is important for the OIDs to be unique and organized. Directory Server itself does not manage OIDs, but there are some best practices described in Section 12.2, "Managing Object Identifiers".

2. Create the new attributes. Attribute definitions require a name, a syntax (the allowed format of the values), an OID, and a description of whether the attribute can only be used once per entry or multiple times.

3. Create an object class to contain the new attributes. An object class lists the required attributes for that entry type and the allowed (permissible) attributes. Because the default schema should never be altered, if any new attributes are created, then they should be added to a custom object class.

The schema elements should be planned in advance; do not use multiple attributes for the same information. Whenever possible, use the standard Directory Server schema. Directory Server has hundreds of attributes and dozens of object classes defined in the default schema files. The *Red Hat Directory Server 10 Configuration, Command, and File Reference* lists and describes the standard attributes and object classes; all of the schema can be viewed in the Directory Server Console or read in the schema files in **/usr/share/dirsrv/schema/**. Become familiar with the available schema; then plan what information attributes are missing and how best to fill those gaps with custom attributes. Planning the schema is covered in the *Deployment Guide*.

WARNING

The default object classes and attributes in Directory Server are based on LDAP and X.500 standards and RFCs. Using standard schema makes the Directory Server more easily integrated with other applications and servers and allows interoperability with LDAP clients, legacy Directory Server instances, and future release. It is inadvisable for you to edit the standard attributes or change the object classes.

Keep the following rules in mind when customizing the Directory Server schema:

- Keep the schema as simple as possible.
- Reuse existing schema elements whenever possible.
- Minimize the number of mandatory attributes defined for each object class.
- Do not define more than one object class or attribute for the same purpose.
- Do not modify any existing definitions of attributes or object classes.

NOTE

Never delete or replace the standard schema. Doing so can lead to compatibility problems with other directories or other LDAP client applications.

The schema is loaded into the Directory Server instance when the instance is started; any new schema files are not loaded until the Directory Server is restarted or unless a reload task is initiated. The default custom schema file for an instance, *99user.ldif*, is loaded as the last schema file. If it contains definitions already present in standard schema files, the custom definition will override the standard ones.

12.1.5. Schema Replication

When the directory schema is updated in the **cn=schema** sub-tree, Directory Server stores the changes in the local **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** file, including a change state number (CSN). The updated schema is not automatically replicated to other replicas. The schema replication starts when directory content is updated in the replicated tree. For example, if you update a user or group entry after modifying the schema, the supplier compares the CSN stored in the ***nsSchemaCSN*** attribute with the one on the consumer. If the remote CSN is lower than the one on the supplier, the schema is replicated to the consumer. For a successful replication, all object classes and attribute types on the supplier must be a superset of the consumer's definition.

Example 12.3. Schema subsets and supersets

- On **server1**, the **demo** object class allows the ***a1***, ***a2***, and ***a3*** attributes.
- On **server2**, the **demo** object class allows the ***a1*** and ***a3*** attributes.

In Example 12.3, "Schema subsets and supersets" , the schema definition of the **demo** object class on

server1 is a superset of the object class on **server2**. During the validation phase, when the schema is being replicated or accepted, Directory Server retrieves the superset definitions. For example, if a consumer detects that an object class in the local schema allows less attributes than the object class in the supplier schema, the local schema is updated.

If the schema definitions are successfully replicated, the ***nsSchemaCSN*** attributes are identical on both servers and no longer compared at the beginning of a replication session.

In the following scenarios, the schema is not replicated:

- The schema on one host is a subset of the schema of another host.

  For example, in Example 12.3, "Schema subsets and supersets" , the schema definition of the **demo** object class on **server2** is a subset of the object class on **server1**. Subsets can also occur for attributes (a single-value attribute is a subset of a multi-value attribute) and attribute syntaxes (**IA5** is a subset of **Octet_string**).

- When definitions in supplier schema and consumer schema need to be merged.

  Directory Server does not support merging schemas. For example, if an object class on one server allows the ***a1***, ***a2***, and ***a3*** attributes and ***a1***, ***a3***, and ***a4*** on the other, the schemas are not subsets and cannot be merged.

- Schema files other than **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** are used.

  Directory Server enables you to add additional schema files in the **/etc/dirsrv/slapd-*instance_name*/schema/** directory. However, only the CSN in the **99user.ldif** file is updated. For this reasons, other schema file are only used locally and are not automatically transferred to replication partners. Copy the updated schema file manually to the consumers and reload the schema. For details, see Section 12.7, "Dynamically Reloading Schema" .

  To avoid duplicate schema definitions and to enable automatic replication, store all custom schema in the **/etc/dirsrv/slapd-*instance_name*/schema/99user.ldif** file. For further information about creating custom schema files, see Section 12.6, "Creating Custom Schema Files".

12.2. MANAGING OBJECT IDENTIFIERS

Each LDAP object class or attribute must be assigned a unique name and *object identifier* (OID). An OID is a dot-separated number which identifies the schema element to the server. OIDs can be hierarchical, with a base OID that can be expanded to accommodate different branches. For example, the base OID could be **1**, and there can be a branch for attributes at **1.1** and for object classes at **1.2**.

NOTE

It is not required to have a numeric OID for creating custom schema, but Red Hat strongly recommends it for better forward compatibility and performance.

OIDs are assigned to an organization through the Internet Assigned Numbers Authority (IANA), and Directory Server does not provide a mechanism to obtain OIDs. To get information about obtaining OIDs, visit the IANA website at http://www.iana.org/cgi-bin/enterprise.pl.

After obtaining a base OID from IANA, plan how the OIDs are going to be assigned to custom schema elements. Define a branch for both attributes and object classes; there can also be branches for matching rules and LDAP controls.

Once the OID branches are defined, create an OID registry to track OID assignments. An OID registry is a list that gives the OIDs and descriptions of the OIDs used in the directory schema. This ensures that no OID is ever used for more than one purpose. Publish the OID registry with the custom schema.

12.3. DIRECTORY SERVER ATTRIBUTE SYNTAXES

The attribute's syntax defines the format of the values which the attribute allows; as with other schema elements, the syntax is defined for an attribute using the syntax's OID in the schema file entry. In the Directory Server Console, the syntax is referenced by its friendly name.

The Directory Server uses the attribute's syntax to perform sorting and pattern matching on entries.

For more information about LDAP attribute syntaxes, see RFC 4517.

Supported LDAP attribute syntaxes are covered in section *Directory Server Attribute Syntaxes* of the Red Hat Directory Server 10 Configuration, Command, and File Reference .

12.4. MANAGING CUSTOM SCHEMA IN THE CONSOLE

The Directory Server Console shows all attributes in the schema, and custom attributes can be created, edited, and deleted from the schema.

- Section 12.4.1, “Viewing Attributes and Object Classes”
- Section 12.4.2, “Creating Attributes”
- Section 12.4.3, “Creating Object Classes”
- Section 12.4.4, “Editing Custom Schema Elements”
- Section 12.4.5, “Deleting Schema”

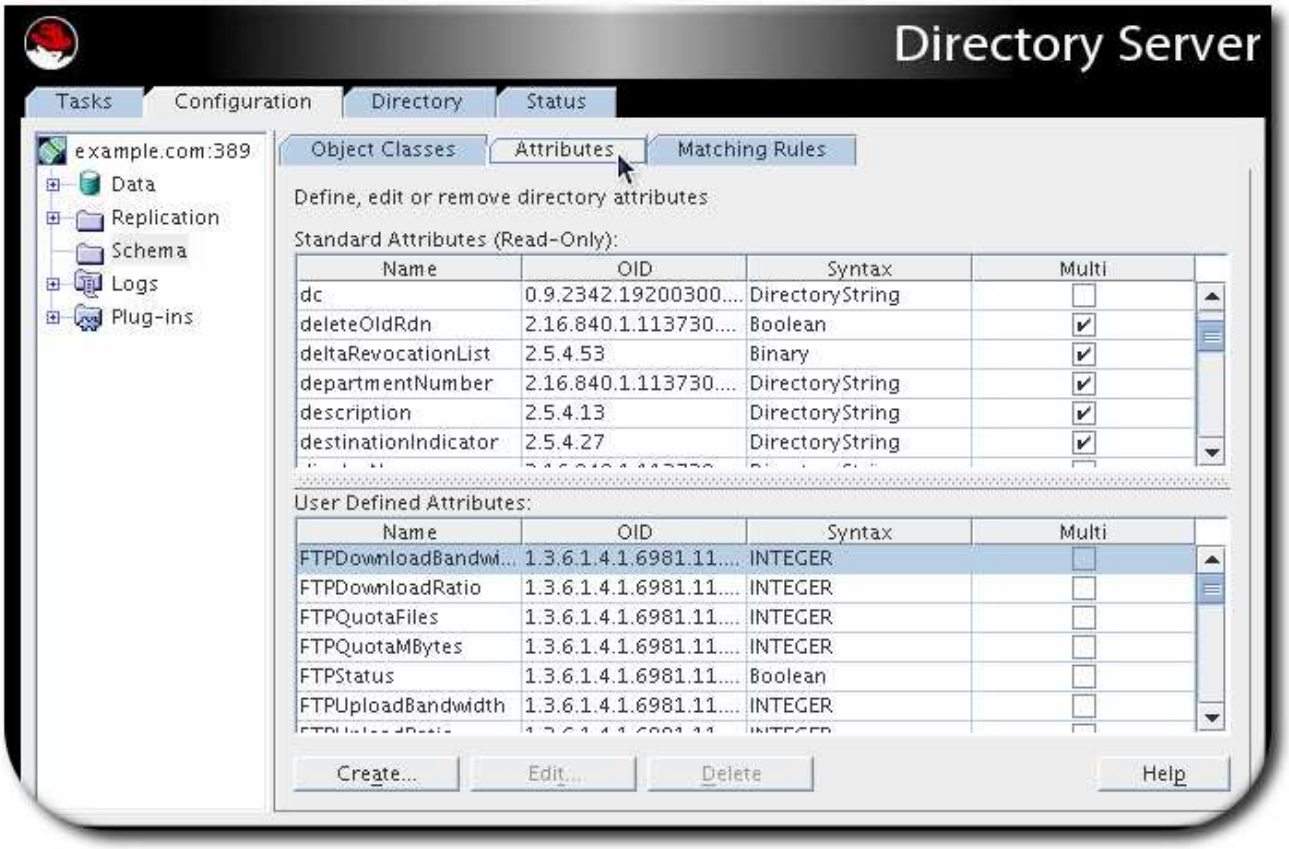
12.4.1. Viewing Attributes and Object Classes

All of the information about the attributes and object classes which are currently loaded in the server instance are visible with the other server configuration.

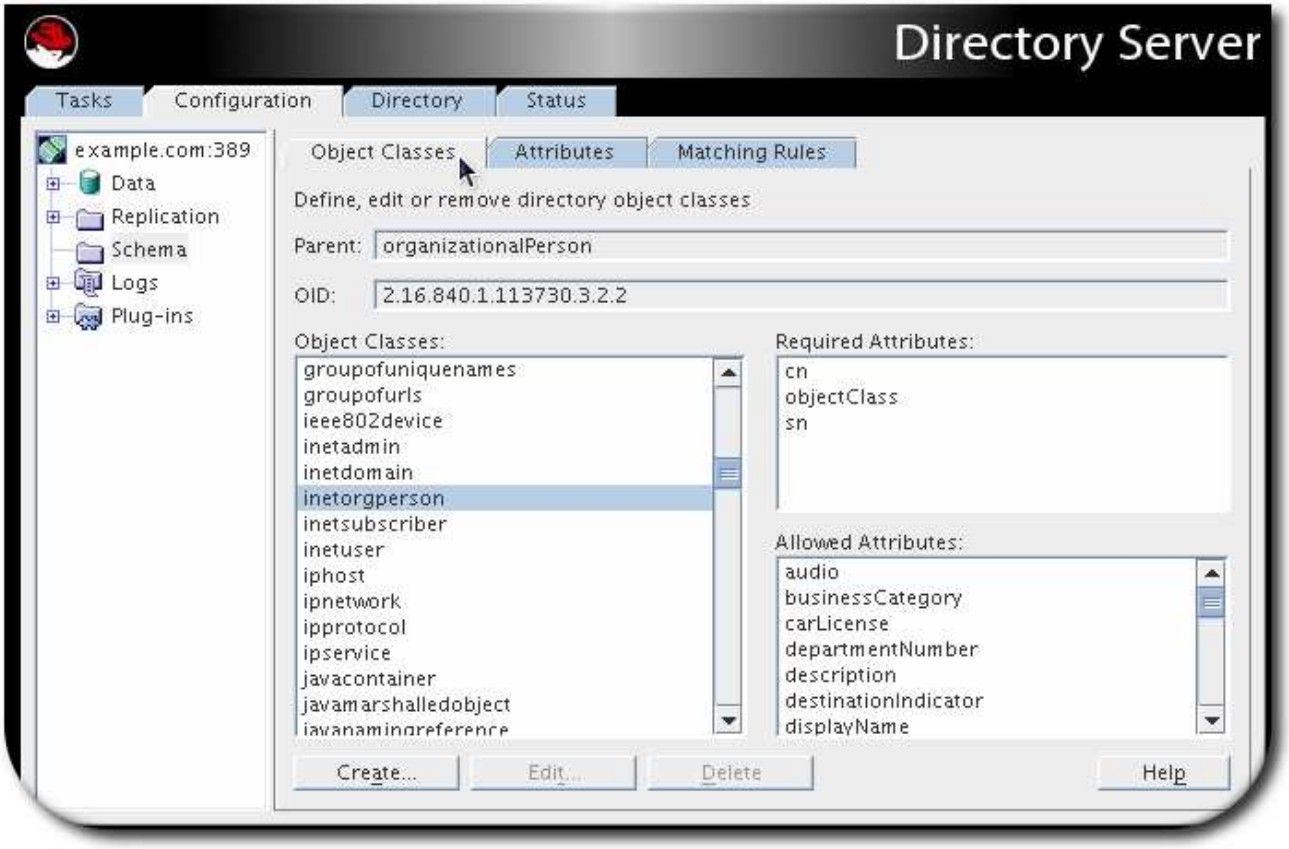
1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder.

3. There are three tabs which display the schema elements loaded in Directory Server: **Object Class**, **Attributes**, and **Matching Rules**.

The **Attributes** tab is broken into two sections for default and custom attributes. Both sections show the attribute name, OID, syntax, and whether the attribute is multi-valued.



The **Object Classes** tab shows the list of object classes on the left. When an object class is highlighted, its OID and superior object class are listed in the fields at the top and its required and allowed attributes are listed in the boxes on the right.

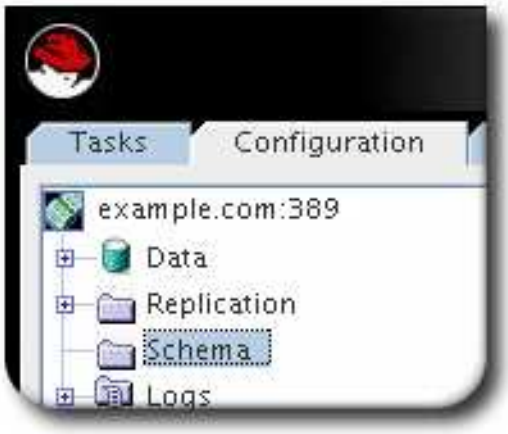


12.4.2. Creating Attributes

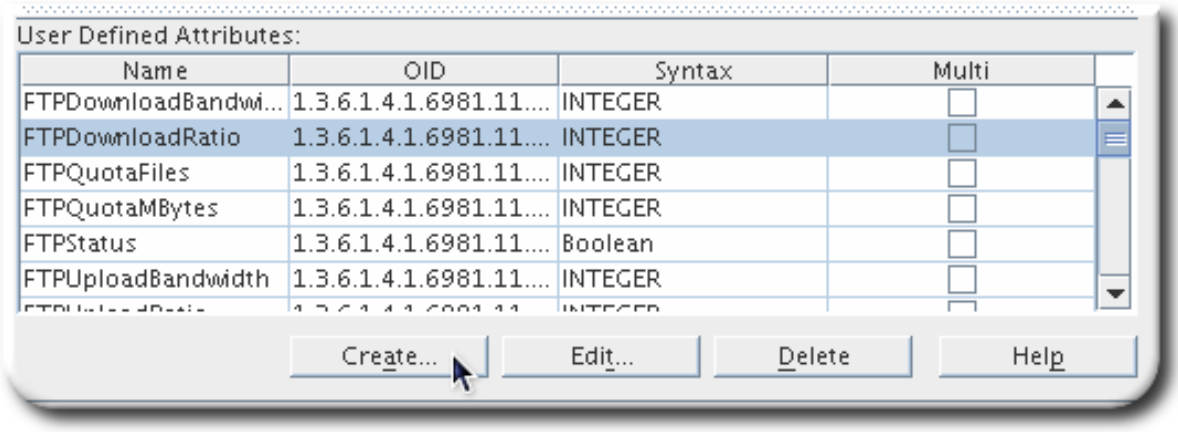
NOTE

After adding new attributes to the schema, create a new object class to contain them, as described in Section 12.4.3, "Creating Object Classes".

1. Select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder, and then select the **Attributes** tab in the right pane.



3. Click **Create**.



4. Fill in the information for the new attribute.

   - The attribute name; this must be unique.
   - The OID; this is not required, but for compatibility and server performance, assigning a unique numeric OID is strongly recommended.
   - The syntax; this is the allowed format for the attributes values.
   - Whether the attribute is multi-valued; by default, all attributes can be used more than once in an entry, but deselecting the check box means the attribute can be used only once.

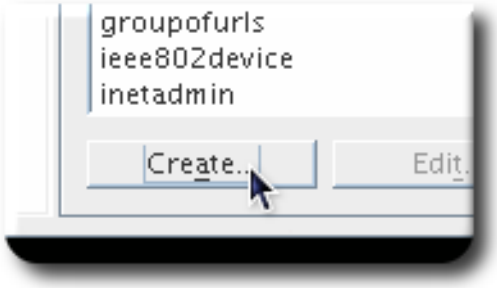
5. Click **OK**.

12.4.3. Creating Object Classes

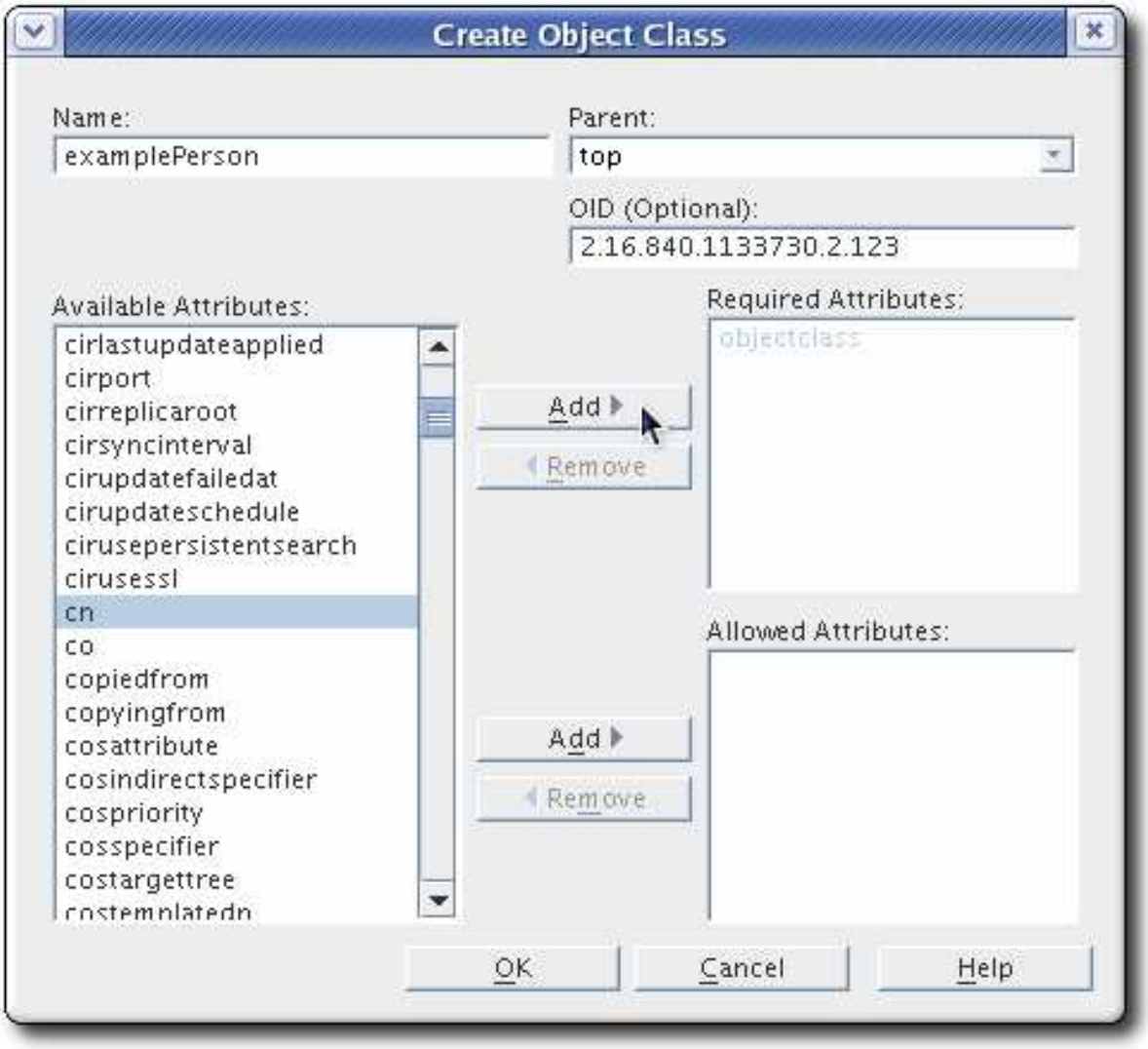
A new object class must be created with a unique name, a parent object, and required and optional attributes. To create an object class:

1. In the Directory Server Console, select the **Configuration** tab.
2. In the navigation tree, select the **Schema** folder, and then select the **Object Classes** tab in the right pane.

3. Click the **Create** button in the **Object Classes** tab.



4. Fill in the information about the new object class.



- The name; this must be unique.
- The OID; this is not required, but for compatibility and server performance, assigning a unique numeric OID is strongly recommended.
- The superior object class for the entry. The default is **top**; selecting another object class means that the new object class inherits all of the required and allowed attributes from the parent, in addition to its own defined attributes.
- Required and allowed attributes. Select the attributes on the left and used the **Add** buttons by the **Available Attributes** and **Required Attributes** boxes to add the attributes as appropriate.

NOTE

Attributes that are inherited from the parent object classes cannot be removed, regardless of whether they are allowed or required.

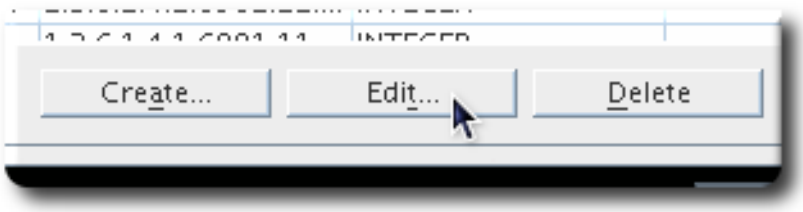
5. Click **OK** to save the new object class.

12.4.4. Editing Custom Schema Elements

Only user-created attributes or object classes can be edited; standard schema elements cannot be edited.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder.

3. Open the **Object Classes** or **Attributes** tab.
4. Select the schema element to edit from the list. Only custom (user-defined) schema can be edited in the Directory Server Console.
5. Click the **Edit** button at the bottom of the window.



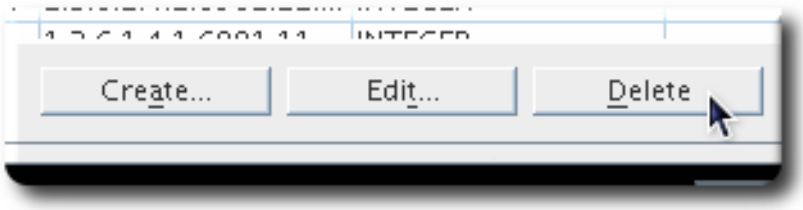
6. Edit any of the schema information.

12.4.5. Deleting Schema

Only user-created attributes or object classes can be deleted; standard schema elements cannot be deleted.

1. In the Directory Server Console, select the **Configuration** tab.
2. In the left navigation tree, select the **Schema** folder.

3. Open the **Object Classes** or **Attributes** tab.
4. Select the schema element to delete from the list. Only custom (user-defined) schema can be deleted in the Directory Server Console.
5. Click the **Delete** button at the bottom of the window.



6. Confirm the deletion.

WARNING

The server immediately deletes the schema element. There is no undo.

12.5. MANAGING SCHEMA USING LDAPMODIFY

As with the Directory Server Console, **ldapmodify** can be used to add, edit, and delete custom schema elements. **ldapmodify** also modifies the default custom schema file for a Directory Server instance, **99user.ldif**.

12.5.1. Creating Attributes

A custom attribute entry is itself an ***attributetypes*** entry for the **cn=schema** entry. The ***attributetypes*** attribute has the format:

```
attributetypes: ( definition )
```

The definition contains five components:

- An OID, usually a dot-separated number

- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The OID for the syntax of the attribute values, discussed in Section 12.3, "Directory Server Attribute Syntaxes", in the form **SYNTAX** *OID*
- Optionally, the source where the attribute is defined

The attribute definition is added to the custom schema file, **99user.ldif**, by running an LDAP command and modifying the **cn=schema** entry. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -x -v

dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.2.3.4.5.6.1 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUED X-ORIGIN 'Example defined')
```

12.5.2. Creating Object Classes

An object class definition is an ***objectclasses*** attribute for the **cn=schema** entry. The ***objectclasses*** attribute has the format:

```
objectclasses: ( definition )
```

The object class definition contains several components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The superior, or parent, object class for this object class, in the form **SUP** *object_class*; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry
- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)
- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)

The object class definition is added to the custom schema file, **99user.ldif**, by by running an LDAP command and modifying the **cn=schema** entry. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -v

dn: cn=schema
changetype: modify
add: objectclasses
```

```
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object
Class' SUP inetOrgPerson AUXILIARY  MUST cn MAY (exampleDateOfBirth $
examplePreferredOS) )
```

12.5.3. Deleting Schema

WARNING

Never delete default schema elements. Those are required by the Directory Server to run.

1. Remove the unwanted attributes from any entries which use them, then from any object classes in the schema file which accept that attribute. Likewise, to remove an object class, remove it from any entries.
2. Run **ldapmodify** to remove the attribute. For example:

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

dn: cn=schema
changetype: modify
delete: objectclasses
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person
Object Class' SUP inetOrgPerson AUXILIARY MUST cn MAY (exampleDateOfBirth $
examplePreferredOS) )
```

WARNING

Be sure to specify the exact object class or attribute to remove; using only the ***attributetypes*** or ***objectclasses*** attribute without the value will delete every user-defined attribute or object class in the file.

If the custom attribute or object class is in a custom schema file other than **99user.ldif**, edit the file directly. Neither the Directory Server Console nor LDAP tools can edit a schema file other than **99user.ldif**.

12.6. CREATING CUSTOM SCHEMA FILES

Schema files are simple LDIF files which define the **cn=schema** entry. Each attribute and object class is added as an attribute to that entry. Here are the requirements for creating a schema file:

- The first line must be **dn: cn=schema**.

- The schema file can include both attributes and object classes, but it can also include only one or the other.
- If both attributes and object classes are defined in the style, all of the attributes must be listed in the file first, then the object classes.
- The object classes can use attributes defined in other schema files.
- The file must be named in the format **[1-9][0-9]text.ldif**.

  The file must always begin with two numbers. Numerically, the schema file cannot be loaded before the core configuration schema (which begin with **00** and **01**).

  Also, the Directory Server always writes its custom schema to the numerically and alphabetically highest named schema file in the schema directory. It expects this file to be **99user.ldif**. If this file is not **99user.ldif**, the server can experience problems. So, always make sure custom schema files are at least alphabetically lower than **99user.ldif**. The name **99alpha.ldif** is okay; the name **99zzz.ldif** is not.

Practices for creating schema files are described in more detail in the *Deployment Guide*.

Attributes are defined in the schema file as ***attributetypes*** attributes to the schema, with five components:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The OID for the syntax of the attribute values, discussed in Section 12.3, "Directory Server Attribute Syntaxes", in the form **SYNTAX** *OID*
- Optionally, the source where the attribute is defined

For example:

```
attributetypes: ( 1.2.3.4.5.6.1 NAME 'dateofbirth' DESC 'For employee birthdays' SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUED X-ORIGIN 'Example defined')
```

Likewise, object classes are defined as values of ***objectclasses*** attributes, although there is slightly more flexibility in how the object class is defined. The only required configurations are the name and OID for the object class; all other configuration depends on the needs for the object class:

- An OID, usually a dot-separated number
- A unique name, in the form **NAME** *name*
- A description, in the form **DESC** *description*
- The superior, or parent, object class for this object class, in the form **SUP** *object_class*; if there is no related parent, use **SUP top**
- The word **AUXILIARY**, which gives the type of entry to which the object class applies; **AUXILIARY** means it can apply to any entry

- A list of required attributes, preceded by the word **MUST**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)
- A list of allowed attributes, preceded by the word **MAY**; to include multiple attributes, enclose the group in parentheses and separate with attributes with dollar signs ($)

For example:

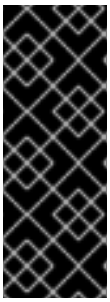
```
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object
Class' SUP inetOrgPerson AUXILIARY  MUST cn MAY (exampleDateOfBirth $
examplePreferredOS) )
```

Example 12.4, "Example Schema File" shows a simplified schema file.

Example 12.4. Example Schema File

```
dn: cn=schema
attributetypes: ( 2.16.840.1133730.1.123 NAME 'dateofbirth' DESC 'For employee birthdays'
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 X-ORIGIN 'Example defined')
objectclasses: ( 2.16.840.1133730.2.123 NAME 'examplePerson' DESC 'Example Person Object
Class' SUP inetOrgPerson AUXILIARY MAY (dateofbirth) )
```

Custom schema files should be added to the Directory Server instance's schema directory, **/etc/dirsrv/slapd-*instance*/schema**. The schema in these files are not loaded and available to the server unless the server is restarted or a dynamic reload task is run.



IMPORTANT

If you want to use a standard schema from the **/usr/share/data/** directory, copy the schema file to the **/usr/share/dirsrv/schema/** directory. If you require that a standard schema is only available to a specific instance, copy the schema file to the **/etc/dirsrv/slapd-*instance_name*/schema/** directory, but use a different file name in the destination directory. Otherwise, Directory Server renames the file during an upgrade and appends the **.bak** suffix.

12.7. DYNAMICALLY RELOADING SCHEMA

By default, the schema files used by the Directory Server instance are loaded into the directory when it is started. This means that any new schema files added to the schema directory are not available for use unless the server is restarted. The Directory Server has a task which manually reloads the full schema for the Directory Server instance, including custom files, without requiring a server restart.

The schema reload task can be initiated in two ways:

- Using the **schema-reload.pl** script
- Adding a **cn=schema reload** task entry using **ldapmodify**

12.7.1. Reloading Schema Using schema-reload.pl

The **schema-reload.pl** script launches a special task to reload all of the schema files used by a specific Directory Server instance. This allows custom schema files to be loaded dynamically without having to add schema elements to **99user.ldif**.

1. Run the script, binding as the Directory Manager.

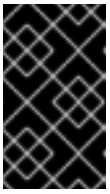
```
# schema-reload.pl -Z instance_name -D "cn=Directory Manager" -w secret
```

The Directory Server responds that it has added the new reload task entry.

```
adding new entry cn=schema_reload_2009_1_6_17_52_4,cn=schema reload
task,cn=tasks,cn=config
```

This reloads the schema from the default schema directory, **/etc/dirsrv/slapd-*instance*/schema**, which is recommended. It is also possible to specify a different directory using the **-d** option.

```
# schema-reload.pl -Z instance_name-D "cn=Directory Manager" -w password -d
/export/custom-schema
```



IMPORTANT

The Directory Server schema reload task reloads the schema files from the directory you specified in the ***schemadir*** parameter. Additionally, the server loads all schema files from the **//usr/share/dirsrv/schema** directory.

The **schema-reload.pl** is described in more detail in the *Configuration, Command, and File Reference*.

12.7.2. Reloading Schema Using ldapmodify

The **schema-reload.pl** script creates a special task entry in a Directory Server instance which reloads schema files; it is also possible to reload schema by creating the task entry directly. Task entries occur under the **cn=tasks** configuration entry in the **dse.ldif** file, so it is also possible to initiate a task by adding the entry using **ldapmodify**. As soon as the task is complete, the entry is removed from the directory.

To initiate a schema reload task, add an entry under the **cn=schema reload task,cn=tasks,cn=config** entry. The only required attribute is the ***cn*** for the specific task.

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=example schema reload,cn=schema reload task,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn:example schema reload
```

The default schema directory from which the Directory Server instance reloads the schema is in **/usr/share/dirsrv/schema**; it is possible to specify a different schema directory using the ***schemadir*** attribute, which is analogous to the **-d** option with **schema-reload.pl**.

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=example schema reload,cn=schema reload task,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn:example schema reload
schemadir: /etc/dirsrv/slapd-instance_name/schema/
```

IMPORTANT

The Directory Server schema reload task reloads the schema files from the directory you specified in the ***schemadir*** parameter. Additionally, the server loads all schema files from the **//usr/share/dirsrv/schema** directory.

As soon as the task is completed, the entry is deleted from the **dse.ldif** configuration, so it is possible to reuse the same task entry continually.

The **cn=schema reload task** configuration is described in more detail in the *Configuration, Command, and File Reference*.

12.7.3. Reloading Schema with Replication

The schema reload task is a local operation, so schema changes are not replicated in a multi-master environment if the schema is added to one supplier but not to the others. To load the new schema files on all of the supplier servers:

1. Stop replication.
2. Copy the new schema file over and run the schema reload task for every supplier and replica server.
3. Restart replication.

12.7.4. Schema Reload Errors

When the schema reload task runs, the command prompt only shows that the task is initiated.

```
adding new entry cn=schema reload task 1,cn=schema reload task,cn=tasks,cn=config
```

However, the task does not return whether it completed successfully. To verify the schema reload operation was successful, check the error logs. The schema reload has two tasks, first validating the schema file and then loading it.

A success message shows that the validation passed and the task finished.

```
[06/Jan/2009:17:52:04.001214874 -0500] schemareload - Schema reload task starts (schema dir:
default) ...
[06/Jan/2009:17:52:04.754335904 -0500] schemareload - Schema validation passed.
[06/Jan/2009:17:52:04.894255328 -0500] schemareload - Schema reload task finished.
```

If there is a failure, then the logs show which step failed and why.

```
[..] schemareload - Schema reload task starts (schema dir: /bogus) ...
[..] schema - No schema files were found in the directory /bogus
[..] schema_reload - schema file validation failed
[..] schemareload - Schema validation failed.
```

12.8. TURNING SCHEMA CHECKING ON AND OFF

When schema checking is on, the Directory Server ensures three things:

- The object classes and attributes using are defined in the directory schema.
- The attributes required for an object class are contained in the entry.
- Only attributes allowed by the object class are contained in the entry.

IMPORTANT

Red Hat recommends not to disable the schema checking.

Schema checking is turned on by default in the Directory Server, and the Directory Server should always run with schema checking turned on. The only situation where is may be beneficial to turn schema checking off is to accelerate LDAP import operations. However, there is a risk of importing entries that do not conform to the schema. Consequently, it is impossible to update these entries.

12.8.1. Turning Schema Checking On and Off Using the Command Line

To turn schema checking on and off using LDAP commands, edit the value of the ***nsslapd-schemacheck*** attribute. For example to disable schema checking:

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

dn: cn=config
changetype: modify
replace: nsslapd-schemacheck
nsslapd-schemacheck: off
```

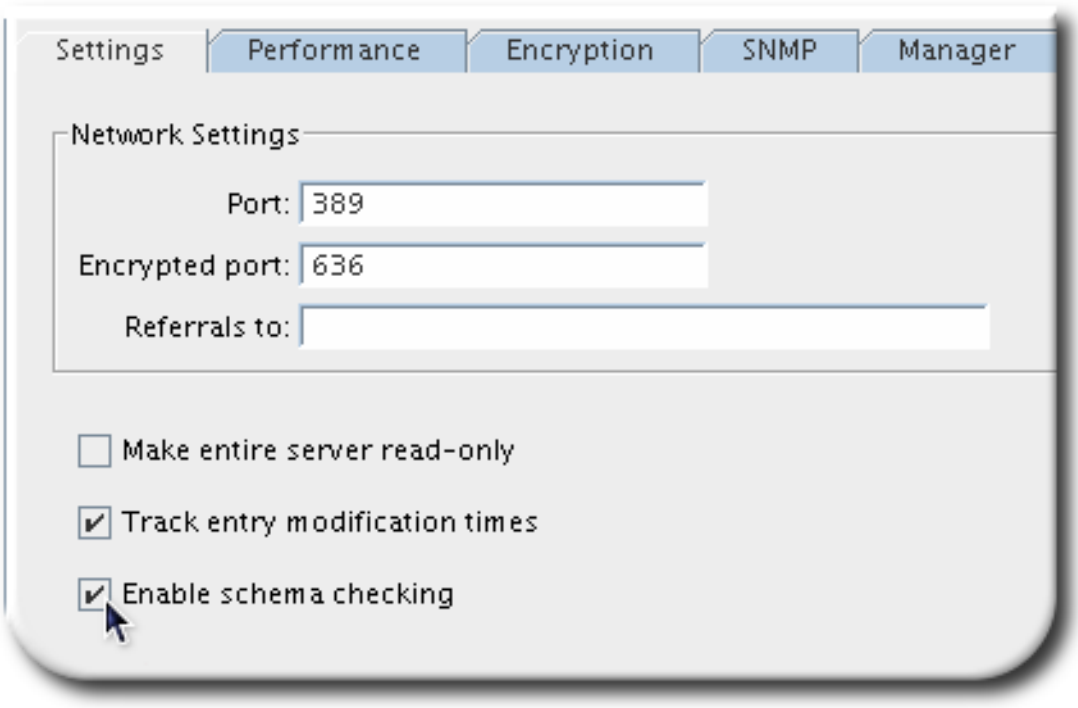
For details about the ***nsslapd-schemacheck*** parameter, see the description of the parameter in the *Red Hat Directory Server Configuration, Command, and File Reference* .

12.8.2. Turning Schema Checking On and Off Using the Console

To enable or disable schema checking using the Console:

1. In the Directory Server Console, select the **Configuration** tab.

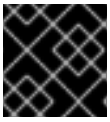
2. Highlight the server icon at the top of the navigation tree, then select the **Settings** tab in the right pane.
3. To enable schema checking, check the **Enable Schema Checking** check box; clear it to turn off schema checking.



4. Click **Save**.

12.9. USING SYNTAX VALIDATION

With syntax validation, the Directory Server checks that the value of an attribute follows the rules of the syntax given in the definition for that attribute. For example, syntax validation will confirm that a new ***telephoneNumber*** attribute actually has a valid telephone number for its value.



IMPORTANT

Red Hat recommends not to disable the syntax validation.

12.9.1. About Syntax Validation

As with schema checking, validation reviews any directory modification and rejects changes that violate the syntax. Additional settings can be optionally configured so that syntax validation can log warning messages about syntax violations and then either reject the modification or allow the modification process to succeed.

This feature validates all attribute syntaxes, with the exception of binary syntaxes (which cannot be verified) and non-standard syntaxes, which do not have a defined required format. The syntaxes are validated against RFC 4514.

12.9.2. Syntax Validation and Other Directory Server Operations

Syntax validation is mainly relevant for standard LDAP operations like creating entries (add) or editing attributes (modify). Validating attribute syntax can impact other Directory Server operations, however.

Database Encryption

For normal LDAP operations, an attribute is encrypted just before the value is written to the database. This means That encryption occurs *after* the attribute syntax is validated.

Encrypted databases (as described in Chapter 10, *Configuring Attribute Encryption*) can be exported

and imported. Normally, it is strongly recommended that these export and import operations are done with the **-E** flag with **db2ldif** and **ldif2db**, which allows syntax validation to occur just fine for the import operation. However, if the encrypted database is exported without using the **-E** flag (which is not supported), then an LDIF with encrypted values is created. When this LDIF is then imported, the encrypted attributes cannot be validated, a warning is logged, and attribute validation is skipped in the imported entry.

Synchronization

There may be differences in the allowed or enforced syntaxes for attributes in Windows Active Directory entries and Red Hat Directory Server entries. In that case, the Active Directory values could not be properly synchronized over because syntax validation enforces the RFC standards in the Directory Server entries.

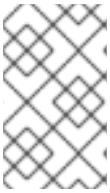
Replication

If the Directory Server 10.6 instance is a supplier which replicates its changes to a consumer, then there is no issue with using syntax validation. However, if the supplier in replication is an older version of Directory Server or has syntax validation disabled, then syntax validation should not be used on the consumer because the Directory Server 10.6 consumer may reject attribute values that the master allows.

12.9.3. Enabling or Disabling Syntax Validation

Syntax validation is configured by the ***nsslapd-syntaxcheck*** attribute. The value of this attribute is either **on** or **off** (by default, this is on). To change the syntax validation, modify this attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-syntaxcheck
nsslapd-syntaxcheck: off
```



NOTE

If syntax validation is disabled, then run the **syntax-validate.pl** script to audit existing attribute values before re-enabling syntax validation. See Section 12.9.6, “Validating the Syntax of Existing Attribute Values”.

12.9.4. Enabling Strict Syntax Validation for DNs

When syntax validation is enabled, DNs are validated against RFC 4514, as are other attribute syntaxes. However, DN syntax validation is enabled separately because the strictness of later standards can invalidate old-style DNs, and therefore directory trees.

Syntax validation checks DNs against section 3 in RFC 4514.

The value of this attribute is either **on** or **off** (by default, this is off). To change the syntax validation, modify this attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
```

```
changetype: modify
replace: nsslapd-dn-validate-strict
nsslapd-dn-validate-strict: on
```

NOTE

If strict DN validation is enabled and a DN value does not conform to the required syntax, then the operation fails with LDAP result code 34, INVALID_DN_SYNTAX.

12.9.5. Enabling Syntax Validation Warnings (Logging)

By default, syntax validation rejects any add or modify operations where an attribute value violates the required syntax. However, the violation itself is not recorded to the errors log by default. The ***nsslapd-syntaxlogging*** attribute enables error logging for any syntax violations.

NOTE

Syntax violations discovered by the syntax validation script and task are logged in the Directory Server error log.

If ***nsslapd-syntaxlogging*** and ***nsslapd-syntaxcheck*** are both enabled, then any invalid attribute modification is rejected and the message written to the log. If ***nsslapd-syntaxlogging*** is enabled but ***nsslapd-syntaxcheck*** is disabled, then the operation is allowed to succeed, but the warning message is still written to the error log.

The value of this attribute is either **on** or **off** (by default, this is off). To enable syntax validation logging, edit the attribute using **ldapmodify** or by editing the **dse.ldif** file directly.

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
replace: nsslapd-syntaxlogging
nsslapd-syntaxlogging: on
```

12.9.6. Validating the Syntax of Existing Attribute Values

In certain situations, you might want to manually validate the syntax of existing values. For example:

- If syntax validation is disabled in the ***nsslapd-syntaxcheck*** parameter. For details, see Section 12.9.3, “Enabling or Disabling Syntax Validation” .

  IMPORTANT

  Red Hat recommends not disabling syntax validation.

- If you migrate data from a server without or disabled syntax validation.

To create a task that validates the syntax of all values in the **ou=people,dc=example,dc=com** sub-tree which match the **(objectclass=inetorgperson)** filter:

```
# syntax-validate.pl -D "cn=Directory Manager" -w secret \
```

```
    -b "ou=people,dc=example,dc=com" -f "(objectclass=inetorgperson)"
ldap_initialize( ldap://server.example.com:389 )
Successfully added task entry "cn=syntax_validate_2017_7_3_10_52_47, cn=syntax validate,
cn=tasks, cn=config"
```

Directory Server logs the results to the **/var/log/dirsrv/slapd-*instance_name*/errors** file. For example:

- If all verified values are valid:

```
[28/Jun/2017:12:52:43.669867966 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Starting (base: "dc=example,dc=com", filter: "(objectclass=*)")
...
[28/Jun/2017:12:52:43.696850129 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Complete.  Found 0 invalid entries.
```

- If invalid entries were found:

```
[28/Jun/2017:12:54:05.736087520 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Starting (base: "dc=example,dc=com", filter: "(objectclass=*)")
...
[28/Jun/2017:12:54:05.754195607 +0200] - ERR - syntax-plugin -
syntax_validate_task_callback - Entry "cn=user,ou=People,dc=example,dc=com" violates
syntax.
description: value #0 invalid per syntax
[28/Jun/2017:12:54:05.759905671 +0200] - ERR - syntax-plugin -
syntax_validate_task_thread - Complete.  Found 1 invalid entries.
```

NOTE

The **syntax-validate.pl** script only identifies syntax violations. You must fix incorrect values manually.

CHAPTER 13. MANAGING INDEXES

Indexing makes searching for and retrieving information easier by classifying and organizing attributes or values. This chapter describes the searching algorithm itself, placing indexing mechanisms in context, and then describes how to create, delete, and manage indexes.

13.1. ABOUT INDEXES

This section provides an overview of indexing in Directory Server. It contains the following topics:

13.1.1. About Index Types

Indexes are stored in files in the directory's databases. The names of the files are based on the indexed attribute, not the type of index contained in the file. Each index file may contain multiple types of indexes if multiple indexes are maintained for the specific attribute. For example, all indexes maintained for the common name attribute are contained in the **cn.db** file.

Directory Server supports the following types of index:

- *Presence index (pres)* contains a list of the entries that contain a particular attribute, which is very useful for searches. For example, it makes it easy to examine any entries that contain access control information. Generating an **aci.db** file that includes a presence index efficiently performs the search for **ACI=*** to generate the access control list for the server.

- *Equality index (eq)* improves searches for entries containing a specific attribute value. For example, an equality index on the ***cn*** attribute allows a user to perform the search for **cn=Babs Jensen** far more efficiently.

- *Approximate index (approx)* is used for efficient approximate or *sounds-like* searches. For example, an entry may include the attribute value **cn=Robert E Lee**. An approximate search would return this value for searches against **cn~=Robert Lee**, **cn~=Robert**, or **cn~=Lee**. Similarly, a search against **l~=San Fransisco** (note the misspelling) would return entries including **l=San Francisco**.

- *Substring index (sub)* is a costly index to maintain, but it allows efficient searching against substrings within entries. Substring indexes are limited to a minimum of three characters for each entry.

  For example, searches of the form **cn=*derson** , match the common names containing strings such as **Bill Anderson**, **Jill Henderson**, or **Steve Sanderson**. Similarly, the search for **telephoneNumber= *555*** returns all the entries in the directory with telephone numbers that contain **555**.

- *International index* speeds up searches for information in international directories. The process for creating an international index is similar to the process for creating regular indexes, except that it applies a *matching rule* by associating an *object identifier* (OID) with the attributes to be indexed.

The supported locales and their associated OIDs are listed in Appendix D, *Internationalization*. If there is a need to configure the Directory Server to accept additional matching rules, contact Red Hat Consulting.

- *Browsing index*, or *virtual list view (VLV) index*, speeds up the display of entries in the Directory Server Console. This index is particularly useful if a branch of your directory contains hundreds of entries; for example, the **ou=people** branch. You can create a browsing index on any branch point in the directory tree to improve display performance through the Directory Server Console or by using the **vlvindex** command-line tool, which is explained in the *Red Hat Directory Server Configuration, Command, and File Reference*.

13.1.2. About Default and Database Indexes

Directory Server contains a set of default indexes. When you create a new database, Directory Server copies these default indexes from **cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config** to the new database. Then the database only uses the copy of these indexes, which are stored in **cn=index,cn=*database_name*,cn=ldbm database,cn=plugins,cn=config**.

NOTE

Directory Server does not replicate settings in the cn=config entry. Therefore, you can configure indexes differently on servers that are part of a replication topology. For example, in an environment with cascading replication, you do not need to create custom indexes on a hub, if clients do not read data from the hub.

To display the Directory Server default indexes:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com \
    -b "cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config" \
    '(objectClass=nsindex)'
```

NOTE

If you update the default index settings stored in **cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config**, the changes are not applied to the individual databases in **cn=index,cn=*database_name*,cn=ldbm database,cn=plugins,cn=config**.

To display the indexes of an individual database:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com \
    -b "cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config" \
    '(objectClass=nsindex)'
```

13.1.3. Overview of the Searching Algorithm

Indexes are used to speed up searches. To understand how the directory uses indexes, it helps to understand the searching algorithm. Each index contains a list of attributes (such as the ***cn***, common name, attribute) and a pointer to the entries corresponding to each value. Directory Serverprocesses a search request as follows:

1. An LDAP client application sends a search request to the directory.

2. The directory examines the incoming request to make sure that the specified base DN matches a suffix contained by one or more of its databases or database links.

    - If they do match, the directory processes the request.
    - If they do not match, the directory returns an error to the client indicating that the suffix does not match. If a referral has been specified in the **nsslapd-referral** attribute under **cn=config**, the directory also returns the LDAP URL where the client can attempt to pursue the request.
    - The Directory Server examines the search filter to see what indexes apply, and it attempts to load the list of entry IDs from each index that satisfies the filter. The ID lists are combined based on whether the filter used AND or OR joins.
    - If the list of entry IDs is larger than the configured ID list scan limit or if there is no index, then the Directory Server searches every entry in the database. This is an *unindexed* search.

3. The Directory Server reads every entry from the **id2entry.db** database or the entry cache for every entry ID in the ID list (or from the entire database for an unindexed search). The server then checks the entries to see if they match the search filter. Each match is returned as it is found.

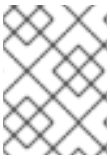
    The server continues through the list of IDs until it has searched all candidate entries or until it hits one of the configured resource limits. (Resource limits are listed in Section 14.1.4, "Setting User and Global Resource Limits Using the Command Line".)

NOTE

It's possible to set separate resource limits for searches using the simple paged results control. For example, administrators can set high or unlimited size and look-through limits with paged searches, but use the lower default limits for non-paged searches.

13.1.4. Approximate Searches

In addition, the directory uses a variation of the metaphone phonetic algorithm to perform searches on an approximate index. Each value is treated as a sequence of words, and a phonetic code is generated for each word.



NOTE

The metaphone phonetic algorithm in Directory Server supports only US-ASCII letters. Therefore, use approximate indexing only with English values.

Values entered on an approximate search are similarly translated into a sequence of phonetic codes. An entry is considered to match a query if both of the following are true:

- All of the query string codes match the codes generated in the entry string.
- All of the query string codes are in the same order as the entry string codes.

Name in the Directory (Phonetic Code)	Query String (Phonetic code)	Match Comments
Alice B Sarette (ALS B SRT)	Alice Sarette (ALS SRT)	Matches. Codes are specified in the correct order.
	Alice Sarrette (ALS SRT)	Matches. Codes are specified in the correct order, despite the misspelling of Sarette.
	Surette (SRT)	Matches. The generated code exists in the original name, despite the misspelling of Sarette.
	Bertha Sarette (BR0 SRT)	No match. The code BR0 does not exist in the original name.
	Sarette, Alice (SRT ALS)	No match. The codes are not specified in the correct order.

13.1.5. Balancing the Benefits of Indexing

Before creating new indexes, balance the benefits of maintaining indexes against the costs.

- Approximate indexes are not efficient for attributes commonly containing numbers, such as telephone numbers.
- Substring indexes do not work for binary attributes.
- Equality indexes should be avoided if the value is big (such as attributes intended to contain photographs or passwords containing encrypted data).
- Maintaining indexes for attributes not commonly used in a search increases overhead without improving global searching performance.
- Attributes that are not indexed can still be specified in search requests, although the search performance may be degraded significantly, depending on the type of search.
- The more indexes you maintain, the more disk space you require.

Indexes can become very time-consuming. For example:

1. The Directory Server receives an add or modify operation.
2. The Directory Server examines the indexing attributes to determine whether an index is maintained for the attribute values.
3. If the created attribute values are indexed, then the Directory Server generates the new index entries.
4. Once the server completes the indexing, the actual attribute values are created according to the client request.

For example, the Directory Server adds the entry:

```
dn: cn=John Doe,ou=People,dc=example,dc=com
objectclass: top
objectClass: person
objectClass: orgperson
objectClass: inetorgperson
cn: John Doe
cn: John
sn: Doe
ou: Manufacturing
ou: people
telephoneNumber: 408 555 8834
description: Manufacturing lead for the Z238 line of widgets.
```

The Directory Server maintains the following indexes:

- Equality, approximate, and substring indexes for ***cn*** (common name) and ***sn*** (surname) attributes.
- Equality and substring indexes for the telephone number attribute.
- Substring indexes for the description attribute.

When adding that entry to the directory, the Directory Server must perform these steps:

1. Create the ***cn*** equality index entry for **John** and **John Doe**.
2. Create the appropriate ***cn*** approximate index entries for **John** and **John Doe**.
3. Create the appropriate ***cn*** substring index entries for **John** and **John Doe**.
4. Create the ***sn*** equality index entry for **Doe**.
5. Create the appropriate ***sn*** approximate index entry for **Doe**.
6. Create the appropriate ***sn*** substring index entries for **Doe**.
7. Create the telephone number equality index entry for **408 555 8834**.
8. Create the appropriate telephone number substring index entries for **408 555 8834**.
9. Create the appropriate description substring index entries for **Manufacturing lead for the Z238 line of widgets**. A large number of substring entries are generated for this string.

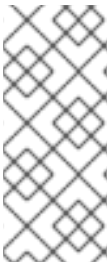
As this example shows, the number of actions required to create and maintain databases for a large directory can be resource-intensive.

13.1.6. Indexing Limitations

You cannot index virtual attributes, such as ***nsrole*** and ***cos_attribute***. Virtual attributes contain computed values. If you index these attributes, Directory Server can return an invalid set of entries to direct and internal searches.

13.2. CREATING STANDARD INDEXES

This section describes how to create presence, equality, approximate, substring, and international indexes for specific attributes using the Directory Server Console and the command line.



NOTE

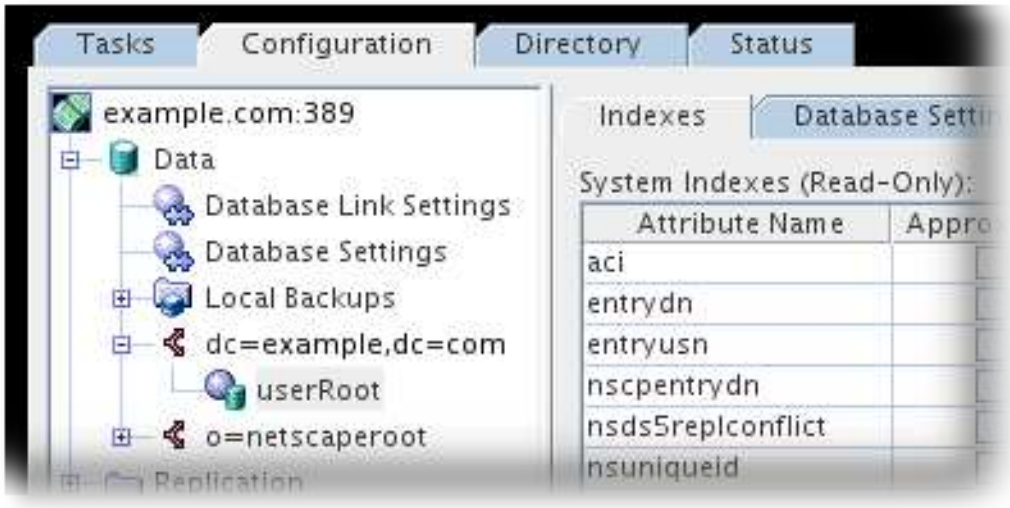
When you create a new index type, Directory Server uses this default index as a template for each new database that will be created in future. If you update the default index, the updated settings are not applied to existing databases. To apply a new index to an existing database, un the **db2index.pl** script or a **cn=index,cn=tasks** task, as described in Section 13.3, "Generating New Indexes to Existing Databases" .

- Section 13.2.1, "Creating Indexes from the Server Console"
- Section 13.2.2, "Creating Indexes from the Command Line"

13.2.1. Creating Indexes from the Server Console

To create presence, equality, approximate, substring, or international indexes:

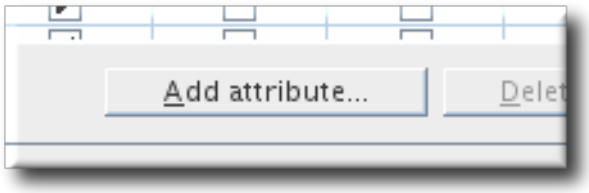
1. Select the **Configuration** tab.
2. Expand the **Data** node, expand the suffix of the database to index, and select the database.
3. Select the **Indexes** tab in the right pane.



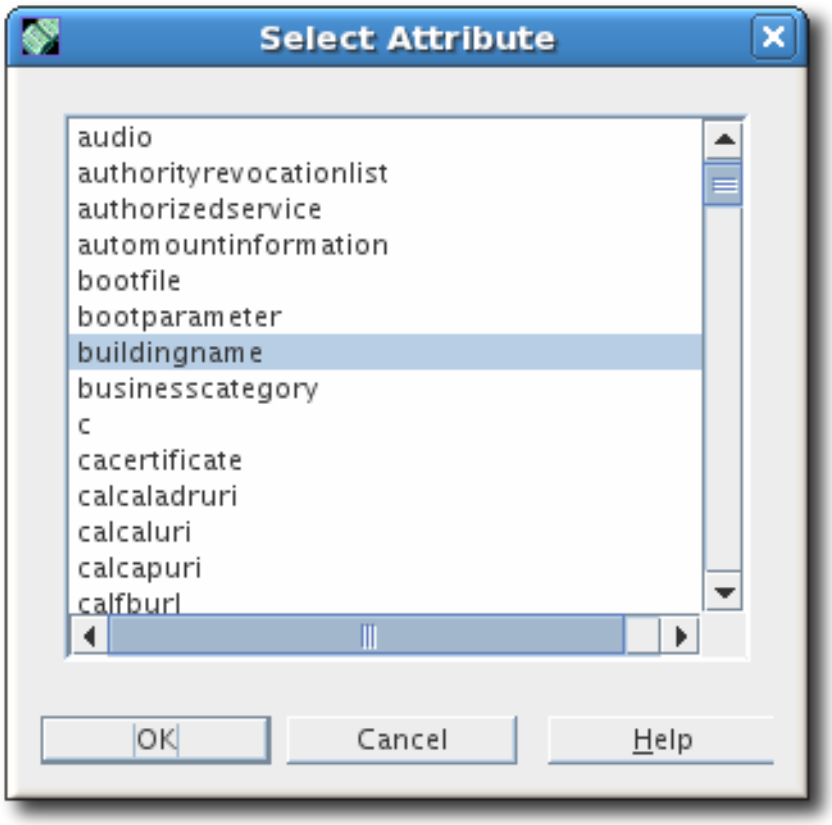
NOTE

Do not click the **Database Settings** node because this opens the **Default Index Settings** window, not the window for configuring indexes per database.

4. If the attribute to be indexed is listed in the **Additional Indexes** table, go to step 6. Otherwise, click **Add Attribute** to open a dialog box with a list of all of the available attributes in the server schema.

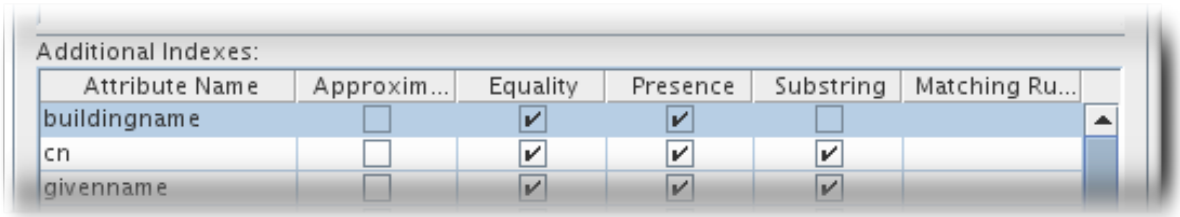


5. Select the attribute to index, and click **OK**.



The server adds the attribute to the **Additional Indexes** table.

6. Select the check box for each type of index to maintain for each attribute.



7. To create an index for a language other than English, enter the OID of the *collation order* to use in the **Matching Rules** field.

   To index the attribute using multiple languages, list multiple OIDs separated by commas, but no whitespace. For a list of languages, their associated OIDs, and further information regarding collation orders, see Appendix D, *Internationalization*.

8. Click **Save**.

The new index is immediately active for any new data that you add and any existing data in your directory. You do not have to restart your server.

13.2.2. Creating Indexes from the Command Line

NOTE

You cannot create new system indexes because system indexes are hard-coded in Directory Server.

Use **ldapmodify** to add the new index attributes to your directory.

- To create a new index that will become one of the default indexes, add the new index attributes to the **cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config** entry.
- To create a new index for a particular database, add it to the **cn=index,cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** entry, where **cn=***database_name* corresponds to the name of the database.

NOTE

Avoid creating entries under **cn=config** in the **dse.ldif** file. The **cn=config** entry in the simple, flat **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will probably suffer. Although we recommend you do not store simple user entries under **cn=config** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **cn=config** since this centralizes configuration information.

For information on the LDIF update statements required to add entries, see Section 3.1.4, "Updating a Directory Entry".

For example, to create presence, equality, and substring indexes for the ***sn*** (surname) attribute in the **Example1** database:

1. Run **ldapmodify** and add the LDIF entry for the new indexes:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
changetype: add
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:pres
nsIndexType:eq
nsIndexType:sub
nsMatchingRule:2.16.840.1.113730.3.3.2.3.1
```

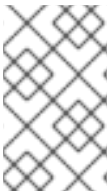
The ***cn*** attribute contains the name of the attribute to index, in this example the ***sn*** attribute. The entry is a member of the **nsIndex** object class. The **nsSystemIndex** attribute is **false**, indicating that the index is not essential to Directory Server operations. The multi-valued **nsIndexType** attribute specifies the presence (**pres**), equality (**eq**) and substring (**sub**)

indexes. Each keyword has to be entered on a separate line. The **nsMatchingRule** attribute in the example specifies the OID of the Bulgarian collation order; the matching rule can indicate any possible value match, such as languages or other formats like date or integer.

You can use the keyword **none** in the **nsIndexType** attribute to specify that no indexes are to be maintained for the attribute. This example temporarily disables the ***sn*** indexes on the **Example1** database by changing the **nsIndexType** to **none**:

```
dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
objectClass:top
objectClass:nsIndex
cn:sn
nsSystemIndex:false
nsIndexType:none
```

For a complete list of matching rules and their OIDs, see Section 14.4.4, “Using Matching Rules”, and for the index configuration attributes, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

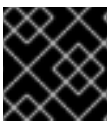


NOTE

Always use the attribute's primary name (not the attribute's alias) when creating indexes. The primary name of the attribute is the first name listed for the attribute in the schema; for example, **uid** for the user ID attribute.

13.3. GENERATING NEW INDEXES TO EXISTING DATABASES

New indexes are not added to existing databases automatically. They must be added manually, and Directory Server has two methods for generating new indexes to an existing database: running the **db2index.pl** script or running a **cn=index,cn=tasks** task.



IMPORTANT

Before you regenerate the index, searches will fail.

13.3.1. Running the db2index.pl Script

After creating an indexing entry or adding additional index types to an existing indexing entry, run the **db2index.pl** script to generate the new set of indexes to be maintained by the Directory Server. After the script is run, the new set of indexes is active for any new data added to the directory and any existing data in the directory.

Run the **db2index.pl** Perl script.

```
# db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n ExampleServer -t sn
```

For more information about using this Perl script, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

For information about the parameters used in the example, see the description of the **db2index** utility in the *Red Hat Directory Server Configuration, Command, and File Reference* .

13.3.2. Using a cn=tasks Entry to Create an Index

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=index,cn=tasks,cn=config** to initiate an indexing operation.

This task entry requires a unique name (***cn***) and a definition for the attribute and type of index, set in ***nsIndexAttribute*** in the format *attribute:index_type*.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=example presence index,cn=index,cn=tasks,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: example presence index
nsInstance: userRoot
nsIndexAttribute: "cn:pres"
```

There are three possible *index_type*s:

- **pres** for presence indexes
- **eq** for equality indexes
- **sub** for substring indexes

As soon as the task is completed, the entry is removed from the directory configuration.

For details about the attributes used in the example and other attributes you can set in this entry, see the **cn=*task_name*,cn=index,cn=tasks,cn=config** entry description in the *Red Hat Directory Server Configuration, Command, and File Reference*.

13.4. CREATING BROWSING (VLV) INDEXES

A *virtual list view (VLV) index* is a way of creating a truncated list for faster searching while enhancing server performance. The VLV index itself can be resource-intensive to maintain, but it can be beneficial in large directories (over 1000 entries).

A browsing index is a type of VLV index that organizes the entries listed into alphabetical order, making it easier to find entries.

VLV indexes are not applied to attributes, like standard indexes are, but they are dynamically generated based on attributes set in entries and the location of those entries in the directory tree. VLV indexes, unlike standard indexes, are special entries in the database rather than configuration settings for the database.

NOTE

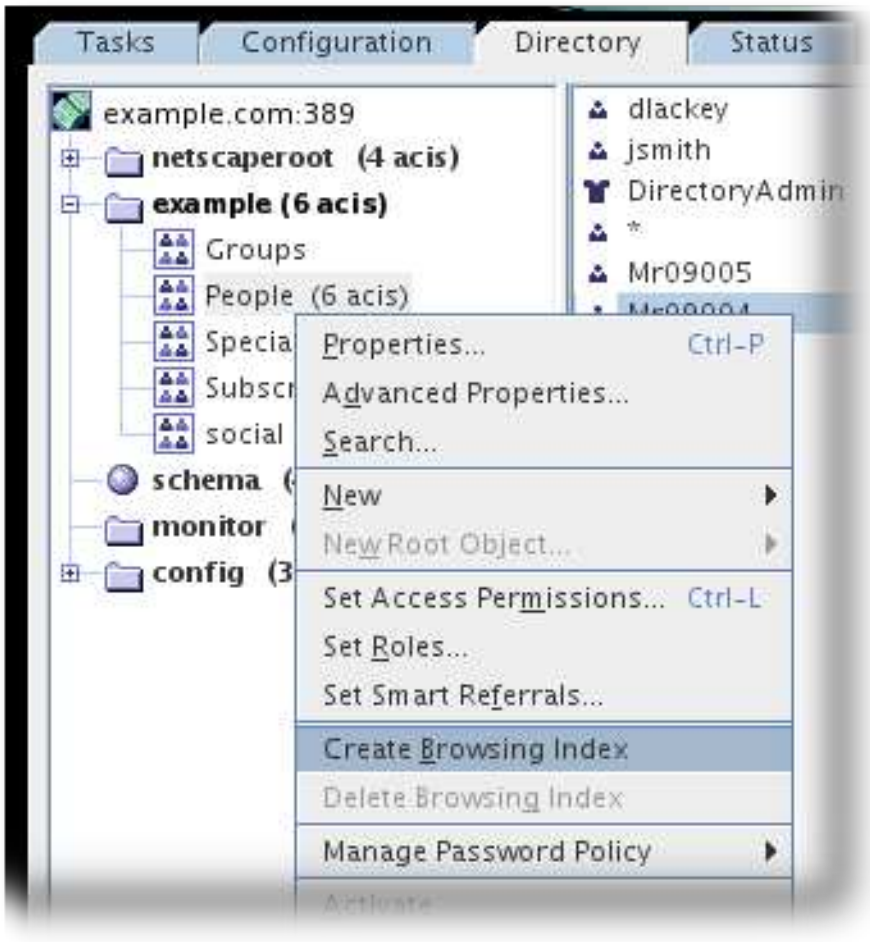
VLV indexes are similar to simple paged results, which can be returned with some external LDAP clients. Simple paged results are calculated per search, while VLV indexes are a permanent list, so VLV indexes are overall faster for searches, but do require some overhead for the server to maintain.

Simple paged results and VLV indexes *cannot* be used on the same search.

For more information, see Section 14.7.4, “Using Simple Paged Results” .

13.4.1. Creating Browsing Indexes from the Server Console

1. Select the **Directory** tab.
2. In the left navigation tree, select the entry, such as **People**, for which to create the index.
3. From the **Object** menu, select **Create Browsing Index**.



The **Create Browsing Index** dialog box appears displaying the status of the index creation. Click the **Status Logs** box to view the status of the indexes created.

4. Click **Close**.

The new index is immediately active for any new data that is added to the directory. You do not have to restart your server.

For more information on how to change the VLV search information or the access control rules that are set by default for VLV searches, see Section 13.4.2.1, "Adding a Browsing Index Entry" and Section 13.4.3, "Setting Access Control for VLV Information".

13.4.2. Creating Browsing Indexes from the Command Line

Creating a browsing index or virtual list view (VLV) index from the command line has these steps:

1. Using **ldapmodify** to add new browsing index entries or edit existing browsing index entries. See Section 13.4.2.1, "Adding a Browsing Index Entry" .

2. Running the **vlvindex** script to generate the new set of browsing indexes to be maintained by the server. See Section 13.4.2.2, "Running the vlvindex Script". Alternatively, launch an appropriate task under **cn=tasks,cn=config** (Section 13.4.2.3, "Using a cn=tasks Entry to Create a Browsing Index").

3. Ensuring that access control on VLV index information is set appropriately. See Section 13.4.3, "Setting Access Control for VLV Information".

13.4.2.1. Adding a Browsing Index Entry

The type of browsing index entry to create depends on the type of **ldapsearch** attribute sorting to accelerate. It is important to take the following into account:

- The scope of the search (base, one, sub)
- The base of the search (the entry to use as a starting point for the search)
- The attributes to sort
- The filter of the search

  For more information on specifying filters for searches, see Chapter 14, *Finding Directory Entries*.

- The LDBM database to which the entry that forms the base of the search belongs. You can only create browsing indexes in LDBM databases.

For example, create a browsing index to accelerate an **ldapsearch** on the entry **ou=People,dc=example,dc=com** held in the **Example1** database with the following attributes:

- The search base is **ou=People,dc=example,dc=com**
- The search filter is **(|(objectclass=*)(objectclass=ldapsubentry))**
- The scope is **one**
- The sorting order for the returned attributes is ***cn***, ***givenname***, ***o***, ***ou***, and ***sn***

1. Run **ldapmodify** and add an entry which specifies the base, scope, and filter of the browsing index:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm
database,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People,dc=example,dc=com
vlvScope: 1
vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))
```

   - The ***cn*** contains the browsing index identifier, which specifies the entry on which to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry. Red Hat recommends using the **dn** of the entry for the browsing index identifier, which is the approach adopted by the Directory Server Console, to prevent identical browsing indexes from being created. The entry is a member of the **vlvSearch** object class.
   - The **vlvbase** attribute value specifies the entry on which you want to create the browsing index; in this example, the **ou=People,dc=example,dc=com** entry (the browsing index identifier).
   - The **vlvScope** attribute is **1**, indicating that the scope for the search you want to accelerate is **1**. A search scope of **1** means that only the immediate children of the entry specified in the ***cn*** attribute, and not the entry itself, will be searched.
   - The **vlvFilter** specifies the filter to be used for the search; in this example, **(|(objectclass=*) (objectclass=ldapsubentry))**.

2. Add the second entry, to specify the sorting order for the returned attributes:

```
dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
  dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,
  cn= config
objectClass: top
objectClass: vlvIndex
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenName o ou sn
```

   - The ***cn*** contains the browsing index sort identifier. The above ***cn*** is the type created by the Console by default, which has the sorting order as being set *by* the browsing index base. The entry is a member of the **vlvIndex** object class.

- The ***vlvSort*** attribute value specifies the order in which you want your attributes to be sorted; in this example, ***cn***, ***givenName***, ***o***, ***ou***, and then ***sn***.

NOTE

This first browsing index entry must be added to the **cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** directory tree node, and the second entry must be a child of the first entry.

13.4.2.2. Running the vlvindex Script

After creating the two browsing indexing entries or added additional attribute types to an existing indexing browsing entries, run the **vlvindex** script to generate the new set of browsing indexes to be maintained by the Directory Server. After running the script, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

To run the **vlvindex** script:

1. Stop the server.

```
# systemctl stop dirsrv@instance_name
```

2. Run the **vlvindex** script.

```
# vlvindex -Z instance_name -n Example1 -T "by MCC ou=people dc=example dc=com"
```

For information about the parameters used in the example, see the description of the **vlvindex** script in the *Red Hat Directory Server Configuration, Command, and File Reference* .

3. Start the server.

```
# systemctl start dirsrv instance
```

13.4.2.3. Using a cn=tasks Entry to Create a Browsing Index

As an alternative to running the **vlvindex** script, it is possible to initiate an indexing task directly.

NOTE

Running the indexing task is the same as running the **vlvindex** script.

The **cn=tasks,cn=config** entry in the Directory Server configuration is a container entry for temporary entries that the server uses to manage tasks. Several common directory tasks have container entries under **cn=tasks,cn=config**. Temporary task entries can be created under **cn=index,cn=tasks,cn=config** to initiate an indexing operation.

This task entry requires a unique name (***cn***) and one other attribute, ***nsIndexVLVAttribute***, which gives the name of the browsing index definition entry to use to generate the VLV index.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
```

```
dn: cn=example VLV index,cn=index,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example VLV index
nsIndexVLVAttribute: "by MCC ou=people,dc=example,dc=com"
```

As soon as the task is completed, the entry is removed from the directory configuration.

The *Red Hat Directory Server Configuration, Command, and File Reference* has more information on running Directory Server tasks under the **cn=tasks** entries.

13.4.3. Setting Access Control for VLV Information

The default access control instruction (ACI) allows only authenticated users to use the VLV index information. If you additionally require to allow non-authenticated users to use the VLV index information, update the ***aci*** attribute to set the ***userdn*** parameter to **ldap:**//**anyone**:

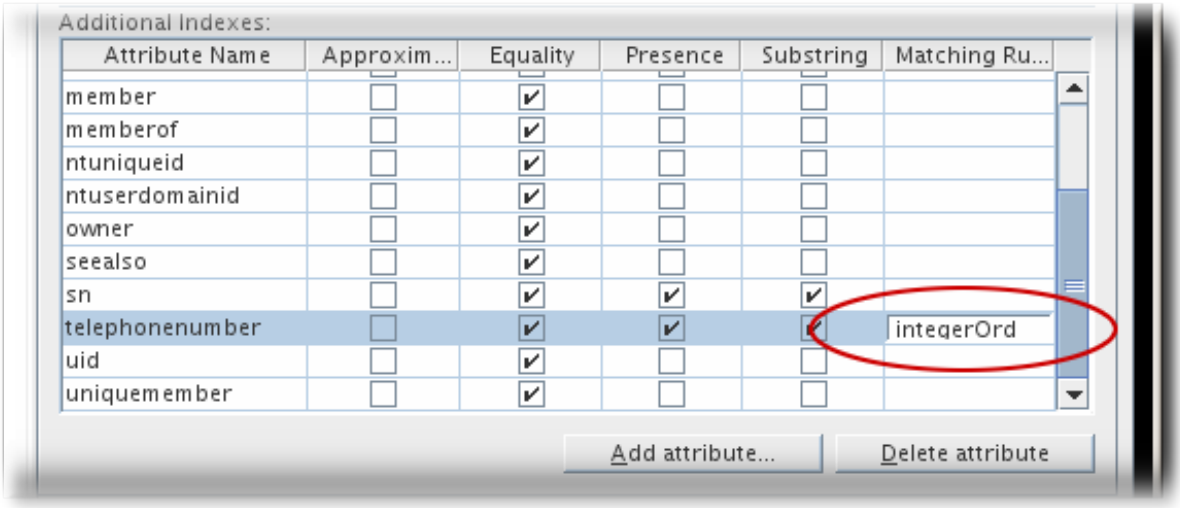
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
changetype: modify
replace: aci
aci: (targetattr != "aci")(version 3.0; acl "VLV Request Control";
    allow( read, search, compare, proxy ) userdn = "ldap://anyone" ;)
```

13.5. CHANGING THE INDEX SORT ORDER

By default, indexes are sorted alphabetically, in descending ASCII order. This is true for every attribute, even attributes which may have numeric attribute values like Integer or TelephoneNumber. It is possible to change the sort method by changing the matching rule set for the attribute.

13.5.1. Changing the Sort Order in the Console

1. Select the **Configuration** tab.
2. Expand the **Data** node, expand the suffix of the database to index, and select the database.
3. Select the **Indexes** tab in the right pane.
4. Select the index, and, in the **Matching Rules** field, enter the new sort order to use. For example, to sort by numbers, rather than alphabetically, enter **integerOrderingMatch**.



5. Click **Save**.

13.5.2. Changing the Sort Order in the Command Line

To change the sort order using the command line, change the ***nsMatchingRule*** for the attribute index. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=sn,cn=index,cn=Example1,cn=ldbm database,cn=plugins,cn=config
changetype:modify
replace:nsMatchingRule
nsMatchingRule:integerOrderingMatch
```

13.6. CHANGING THE WIDTH FOR INDEXED SUBSTRING SEARCHES

By default, for a search to be indexed, the search string must be at least three characters long, without counting any wildcard characters. For example, the string **abc** would be an indexed search while **ab*** would not be. Indexed searches are significantly faster than unindexed searches, so changing the minimum length of the search key is helpful to increase the number of indexed searches.

To improve search performance, particularly for sites with many wildcard searches, the search string length for indexed searches can be changed. Directory Server has three attributes which allow you to change the minimum number of characters required for an indexed search:

- The ***nsSubStrBegin*** attribute sets the required number of characters for an indexed search for the beginning of a search string, before the wildcard.

  ```
  abc*
  ```

- The ***nsSubStrMiddle*** attribute sets the required number of characters for an indexed search where a wildcard is used in the middle of a search string. For example:

  ```
  ab*z
  ```

- The ***nsSubStrEnd*** attribute sets the required number of characters for an indexed search for the end of a search string, after the wildcard. For example:

  -

```
*xyz
```

The default substring search length for the string triplet (before, middle, and end) is 3, 3, and 3, meaning every search requires a minimum of three characters, in every wildcard position.

For any attribute index to have alternate string lengths, add the **extensibleObject** object class to the entry and then set the substring search lengths.

1. Set the new key length for the specific attribute index. This requires adding the **extensibleObject** object class and then adding the ***nsSubStrBegin***, ***nsSubStrEnd***, or ***nsSubStrMiddle*** attributes as appropriate. For example:

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

dn: attribute_name,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
changetype: modify
add: objectclass
objectclass: extensibleObject
-
add: nsSubStrBegin
nsSubStrBegin: 2
-
add: nsSubStrMiddle
nsSubStrMiddle: 2
-
add: nsSubStrEnd
nsSubStrEnd: 2
```

2. Stop the server.

```
# systemctl stop dirsrv.target
```

3. Recreate the attribute index. If even one of the substring search width options is changed, then the entire index must be recreated.

```
# db2index -t attribute_name
```

4. Start the server again.

```
# systemctl start dirsrv.target
```

13.7. DELETING INDEXES

This section describes how to remove attributes and index types from the index.

13.7.1. Deleting an Attribute from the Default Index Entry

When using the default settings of Directory Server, several attributes listed in the default index entry, such as ***sn***, are indexed. The following attributes are part of the default index:

Table 13.1. Default Index Attributes

aci	cn	entryusn
givenName	mail	mailAlternateAddress
mailHost	member	memberOf
nsUniqueId	ntUniqueId	ntUserDomainId
numsubordinates	objectclass	owner
parentid	seeAlso	sn
telephoneNumber	uid	uniquemember

WARNING

Removing system indexes can significantly affect the Directory Server performance.

For example, to remove the ***sn*** attribute from the default index:

1. Remove the attribute from the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** entry:

   ```
   # ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
   cn=sn,cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config
   ```

   If you do not remove the attribute from this entry, the index for the ***sn*** attribute is automatically recreated and corrupted after the server is restarted.

2. Remove the ***cn=attribute_name,cn=index,cn=userRoot,cn=ldbm database,cn=plugins,cn=config*** entry. For details, see:

   - Section 13.7.2, “Removing an Attribute from the Index Using the Server Console”
   - Section 13.7.3, “Removing an Attribute from the Index Using the Command Line”

3. Run the **db2index.pl** Perl script to recreate the index:

   ```
   # db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
   ```

   For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

13.7.2. Removing an Attribute from the Index Using the Server Console

The Directory Server Console can delete any custom indexes, indexes used by other server applications such as a messaging or web server, and default indexes.

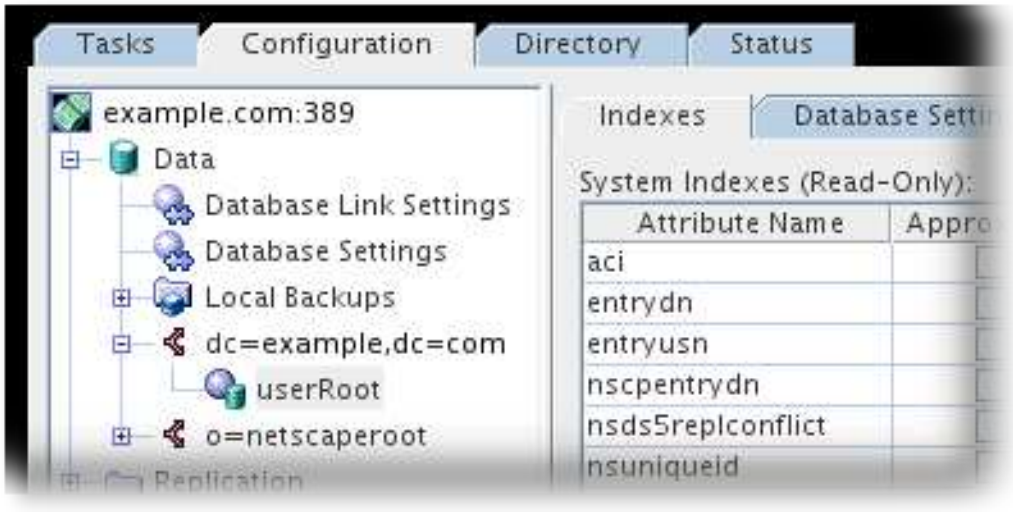
To remove an attribute from the index:

1. If the attribute to remove is listed in the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** default index entry, remove it from this entry first. For details, see Section 13.7.1, "Deleting an Attribute from the Default Index Entry" .

2. Select the **Configuration** tab.

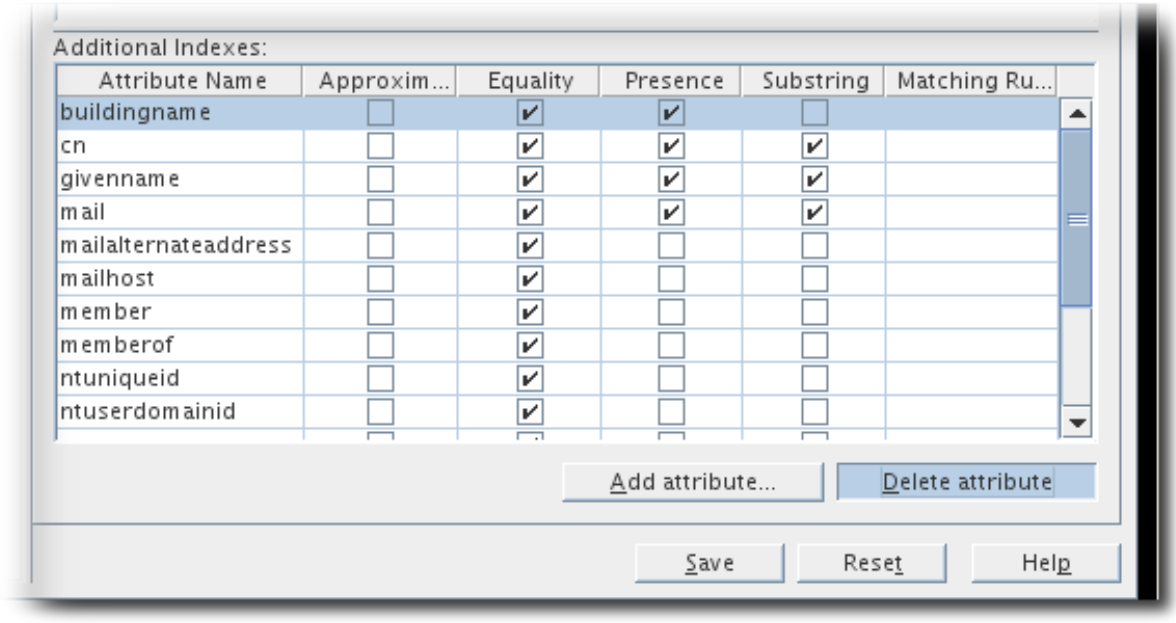
3. Expand the **Data** node and expand the suffix associated with the database containing the index.

4. Select the database from which to delete the index.



5. Locate the attribute containing the index to delete. Clear the check box under the index.

   To delete all indexes maintained for a particular attribute, select the attribute's cell under **Attribute Name**, and click **Delete Attribute**.



6. Click **Save**.

A **Delete Index** warning dialog box opens, requiring a confirmation to delete the index.

7. Click **Yes** to delete the index.

13.7.3. Removing an Attribute from the Index Using the Command Line

In certain situations you want to remove an attribute from the index. For example, to remove the ***sn*** attribute:

1. If the attribute to remove is listed in the ***cn=default indexes,cn=config,cn=ldbm database,cn=plugins,cn=config*** default index entry, you must remove it from this entry first. For details, see Section 13.7.1, “Deleting an Attribute from the Default Index Entry” .

2. Remove the attribute from the index:

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config
```

   After deleting the entry, the index for the ***sn*** attribute is no longer maintained.

3. Run the **db2index.pl** Perl script to recreate the index.

```
# db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
```

   For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

13.7.4. Deleting Index Types from the Command Line

For example, to remove the **sub** index type of the ***sn*** attribute from the index:

1. Remove the index type:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=sn,cn=index,cn=database_name,cn=ldbm database,cn=plugins,cn=config

changetype: modify
delete: nsIndexType
nsIndexType:sub
```

   After deleting the index entry, the substring index for the ***sn*** attribute is no longer maintained.

2. Run the **db2index.pl** Perl script to recreate the index. For example:

```
# db2index.pl -Z instance_name -D "cn=Directory Manager" -w secret -n database_name
```

   For further information about using the **db2index.pl** Perl script, see the db2index.pl(8) man page.

13.7.5. Deleting Browsing Indexes from the Server Console

1. Select the **Directory** tab.

2. Select the entry from which to delete the index in the navigation tree, and select **Delete Browsing Index** from the **Object** menu.

   Alternatively, select and right-click the entry of the index to delete in the navigation tree, and then choose **Delete Browsing Index** from the pop-up menu.

3. A **Delete Browsing Index** dialog box appears asking you to confirm the deletion of the index. Click **Yes**.

4. The **Delete Browsing Index** dialog box appears displaying the status of the index deletion.

13.7.6. Deleting Browsing Indexes from the Command Line

Deleting a browsing index or virtual list view (VLV) index from the command line involves two steps:

1. Using the **ldapdelete** to delete browsing index entries or edit existing browsing index entries (Section 13.7.6.1, "Deleting a Browsing Index Entry").

2. Running the **vlvindex** script to generate the new set of browsing indexes to be maintained by the server (Section 13.7.6.2, "Running the vlvindex Script"). Alternatively, launch an appropriate task under **cn=tasks,cn=config** (Section 13.4.2.3, "Using a cn=tasks Entry to Create a Browsing Index").

The actual entries for an alphabetical browsing index and virtual list view are the same. The following sections describe the steps involved in deleting browsing indexes.

13.7.6.1. Deleting a Browsing Index Entry

Use the **ldapdelete** command-line utility to either delete browsing indexing entries or edit existing browsing index entries. To delete browsing indexes for a particular database, remove the browsing index entries from the **cn=index,cn=***database_name***,cn=ldbm database,cn=plugins,cn=config** entry, where **cn=***database_name* corresponds to the name of the database.

For example, there is a browsing index for accelerating **ldapsearch** operations on the entry **ou=People,dc=example,dc=com**. It held in the **Example1** database where the search base is **ou=People,dc=example,dc=com**, the search filter is **(|(objectclass=*)(objectclass=ldapsubentry))**, the scope is **1**, and the sorting order for the returned attributes is ***cn***, ***givenname***, ***o***, ***ou***, and ***sn***.

To delete this browsing index, delete the two corresponding browsing index entries:

```
dn: cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: vlvSearch
cn: MCC ou=People dc=example dc=com
vlvBase: ou=People,dc=example,dc=com
vlvScope: 1 vlvFilter: (|(objectclass=*)(objectclass=ldapsubentry))

dn: cn=by MCC ou=People dc=example dc=com,cn=MCC ou=People
dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config
objectClass: top
objectClass: vlvIndex
cn: by MCC ou=People dc=example dc=com
vlvSort: cn givenname o ou sn
```

Run **ldapdelete**, specifying both entries.

```
# ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x "cn=MCC ou=People
dc=example dc=com,cn=userRoot,cn=ldbm database,cn=plugins,cn=config" "cn=by MCC ou=People
dc=example dc=com,cn=MCC ou=People dc=example dc=com,cn=userRoot,cn=ldbm
database,cn=plugins,cn=config"
```

After deleting the two browsing index entries, the browsing index will no longer be maintained by the **Example1** database.

13.7.6.2. Running the vlvindex Script

After deleting browsing indexing entries or unwanted attribute types from existing browsing indexing entries, run the **vlvindex** script to generate the new set of browsing indexes to be maintained by the Directory Server. After the script is run, the new set of browsing indexes is active for any new data added to the directory and any existing data in the directory.

1. Stop the server.

   ```
   # systemctl stop dirsrv.target instance
   ```

2. Run the **vlvindex** script.

   ```
   # vlvindex -Z instance_name -n Example1 -T "by MCC ou=people dc=example dc=com"
   ```

   For information about the parameters used in the example, see the description of the **vlvindex** script in the *Red Hat Directory Server Configuration, Command, and File Reference* .

3. Restart the server.

```
# systemctl start dirsrv.target instance
```

Alternatively, create a new task entry under **cn=index,cn=tasks,cn=config** to initiate an indexing operation. This task entry requires a unique name (***cn***) and one other attribute, ***nsIndexVLVAttribute***, which gives the name of the browsing index definition entry to use to generate the VLV index. This task is the same as running **vlvindex**.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=example VLV index,cn=index,cn=tasks,cn=config
changetype: add
objectclass: extensibleObject
cn: example VLV index
nsIndexVLVAttribute: "by MCC ou=people,dc=example,dc=com"
```

As soon as the task is completed, the entry is removed from the directory configuration.

For details about the attributes used in the example and other attributes you can set in this entry, see the **cn=*task_name*,cn=index,cn=tasks,cn=config** entry description in the *Red Hat Directory Server Configuration, Command, and File Reference*.

CHAPTER 14. FINDING DIRECTORY ENTRIES

Entries in the directory can be searched for and found using any LDAP client. Most clients provide some form of search interface so that the directory can be searched easily and entry information can be easily retrieved.

14.1. IMPROVING SEARCH PERFORMANCE THROUGH RESOURCE LIMITS

With large directories, searching through every entry in the database can have a negative impact on the server performance. Effective indexing can improve the performance in certain scenarios. However, in large databases, this may still not reduce the search scope enough to improve the performance.

Reasonable limits can be set on user and client accounts to reduce the total number of entries or the total amount of time spent in an individual search, which both makes searches more responsive and improves overall server performance.

Server limits for search operations are controlled using special operational attribute values on the client application binding to the directory. You can set the following search operation limits:

- *Look through limit.* Specifies how many entries can be examined for a search operation.
- *Size limit.* Specifies the maximum number of entries the server returns to a client application in response to a search operation.
- *Time limit.* Specifies the maximum time the server spends processing a search operation.
- *Idle timeout.* Specifies the time a connection to the server can be idle before the connection is dropped.
- *Range timeout.* Specifies a separate look-through limit specifically for searches using a range.

The resource limits set for the client application take precedence over the default resource limits set for in the global server configuration.

NOTE

The Directory Manager receives unlimited resources by default, with the exception of range searches.

14.1.1. Search Performance and Resource Limits

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

14.1.2. Fine Grained ID List Size

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

14.1.3. Setting Resource Limits on a Single User

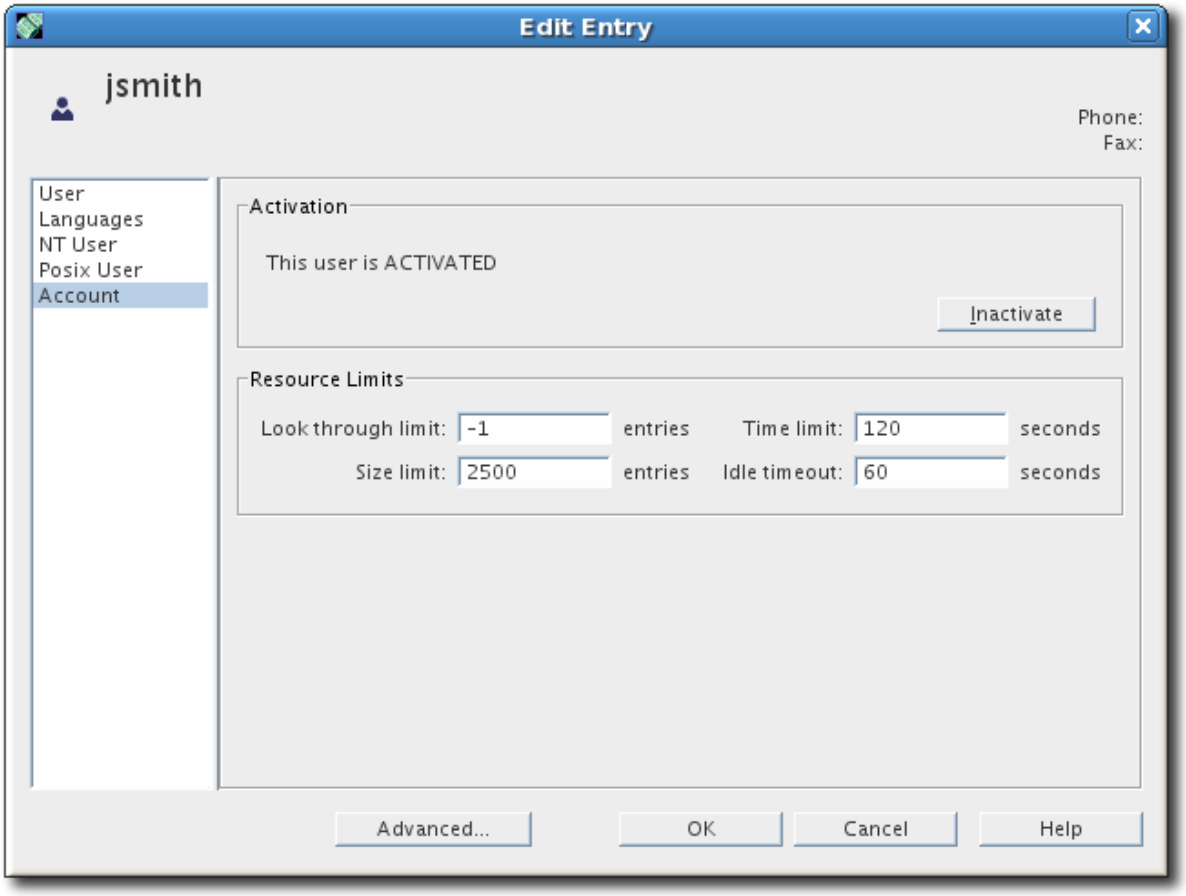
1. Select the **Directory** tab.
2. Browse the navigation tree in the left navigation pane, and double-click the user or role for which to set resource limits.

The **Edit Entry** dialog box appears.

3. Click **Account** in the left pane.

4. Set the resource limits. There are four different limits that can be set:

   - *Look through limit*. The maximum number of entries are examined for a search operation.
   - *Size limit*. The maximum number of entries the server returns to a client application in response to a search operation.
   - *Time limit*. The maximum time the server spends processing a search operation.
   - *Idle timeout*. The time a connection to the server can be idle before the connection is dropped.



Entering a value of **-1** indicates no limit.

5. Click **OK**.

14.1.4. Setting User and Global Resource Limits Using the Command Line

More options are available when setting resource limits in the command line than through the Directory Server Console. The Directory Server Console sets user-level resource limits. Through the command line, administrators can set user-level resource limits, global resource limits, and limits for specific kinds of searches, such as simple paged and range searches. Section 13.1.3, "Overview of the Searching Algorithm" has more information on how these resource limits affect Directory Server search performance.

Section 14.1.4, "Setting User and Global Resource Limits Using the Command Line" lists operational attributes which can be set for each entry using the command line. Use **ldapmodify** to add the attributes to the entry.

User-level attributes are set on the individual entries, while global configuration attributes are set in the appropriate server configuration area.

Look-through limit

Specifies how many entries are examined for a search operation. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsLookThroughLimit***
- Global configuration:
  - Attribute: ***nsslapd-lookthroughlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

Page look-through limit

As with the look-through limit, specifies how many entries are examined, but specifically for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsPagedLookThroughLimit***
- Global configuration:
  - Attribute: ***nsSizeLimit***
  - Entry: **cn=config**

Size limit

Specifies the maximum number of entries the server returns to a client application in response to a search operation. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsSizeLimit***
- Global configuration:
  - Attribute: ***nsslapd-sizelimit***
  - Entry: **cn=config**

Paged size limit

As with the size limit, specifies the maximum number of entries the server returns to a client application but only for simple paged search operations. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsPagedSizeLimit***
- Global configuration:
  - Attribute: ***nsslapd-pagedsizelimit***
  - Entry: **cn=config**

Time Limit

Specifies the maximum time the server spends processing a search operation. Giving this attribute a value of **-1** indicates that there is no time limit.

- User-level attribute: ***nsTimeLimit***
- Global configuration:
  - Attribute: ***nsslapd-timelimit***
  - Entry: **cn=config**

Idle timeout

Specifies the time a connection to the server can be idle before the connection is dropped. The value is given in seconds. Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: ***nsidletimeout***
- Global configuration:
  - Attribute: ***nsslapd-idletimeout***
  - Entry: **cn=config**

ID list scan limit

Specifies the maximum number of entry IDs loaded from an index file for search results. If the ID list size is greater than this value, the search will not use the index list but will treat the search as an unindexed search and look through the entire database.

- User-level attribute: ***nsIDListScanLimit***
- Global configuration:
  - Attribute: ***nsslapd-idlistscanlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

Paged ID list scan limit

As with the ID list scan limit, specifies the maximum number of entry IDs loaded from an index file for search results, but specifically for paged search operations.

- User-level attribute: ***nsPagedIDListScanLimit***
- Global configuration:
  - Attribute: ***nsslapd-pagedidlistscanlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

Range look-through limit

Specifies how many entries are examined for a range search operation (a search using greater-than, equal-to-or-greater-than, less-than, or equal-to-less-than operators). Giving this attribute a value of **-1** indicates that there is no limit.

- User-level attribute: not available
- Global configuration:
  - Attribute: ***nsslapd-rangelookthroughlimit***
  - Entry: **cn=config,cn=ldbm database,cn=plugins,cn=config**

For information about the parameters listed above, see their descriptions in the *Red Hat Directory Server Configuration, Command, and File Reference* .

For example, this sets the size limit for Barbara Jensen by using **ldapmodify** to modify her entry:

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

dn: uid=user_name,ou=People,dc=example,dc=com
changetype: modify
add: nsSizeLimit
nsSizeLimit: 500
```

The **ldapmodify** statement adds the ***nsSizeLimit*** attribute to Babs Jensen's entry and gives it a search return size limit of 500 entries.

NOTE

Set an access control list (ACL) to prevent users changing the setting. For details about ACLs, see Chapter 18, *Managing Access Control*.

14.1.5. Setting Resource Limits on Anonymous Binds

Resource limits are set on a user entry. An anonymous bind, obviously, does not have a user entry associated with it. This means that the global resource limits usually apply to anonymous operations. However, it is possible to configure resource limits specifically for anonymous binds by creating a template user entry that has resource limits, and then applying that template to anonymous binds.

1. Create a template entry and set whatever resource limits you want to apply to anonymous binds.

NOTE

For performance reasons, the template should be in the normal back end, not in the **cn=config** suffix, which does not use an entry cache.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=anon template,ou=people,dc=example,dc=com
changetype: add
objectclass: nsContainer
objectclass: top
cn: anon template
```

```
nsSizeLimit: 250
nsLookThroughLimit: 1000
nsTimeLimit: 60
```

2. On all masters in a replication topology, add the ***nsslapd-anonlimitsdn*** to the server configuration, pointing to the DN of the template entry. Any of the resource limits in Section 14.1.4, "Setting User and Global Resource Limits Using the Command Line" can be set. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=config
changetype: modify
add: nsslapd-anonlimitsdn
nsslapd-anonlimitsdn: cn=anon template,ou=people,dc=example,dc=com
```

14.1.6. Improving Performance for Range Searches

Range searches use operators (Section 14.4.2, "Using Operators in Search Filters") to set a bracket to search for and return an entire subset of entries within the directory. For example, this searches for every entry modified at or after midnight on January 1:

```
(modifyTimestamp>=20200101010101Z)
```

The nature of a range search is that it must evaluate every single entry within the directory to see if it is within the range given. Essentially, a range search is always an all IDs search.

For most users, the look-through limit kicks in and prevents range searches from turning into an all IDs search. This improves overall performance and speeds up range search results. However, some clients or administrative users like Directory Manager may not have a look-through limit set. In that case, a range search can take several minutes to complete or even continue indefinitely.

It is possible to set a separate range look-through limit. This allows clients and administrative users to have high look-through limits while still allowing a reasonable limit to be set on potentially performance-impaired range searches.

This is configured in the ***nsslapd-rangelookthroughlimit*** attribute. The default value is **5000**, the same as the default ***nsslapd-lookthroughlimit*** attribute value.

For example:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=config,cn=ldbm database,cn=plugins,cn=config
changetype: add
add: nsslapd-rangelookthroughlimit
nsslapd-rangelookthroughlimit: 7500
```

14.2. FINDING ENTRIES USING THE DIRECTORY SERVER CONSOLE

Users can browse the **Directory** tab of the Directory Server Console to see the contents of the directory tree and search for specific entries in the directory.

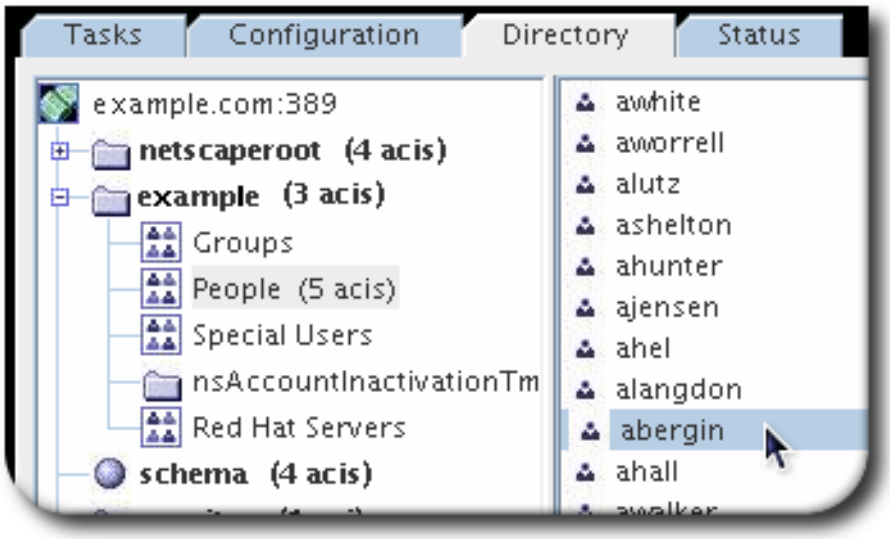


Figure 14.1. Browsing Entries in the Directory Tab

Depending on the DN used to authenticate to the directory, this tab displays the contents of the directory that the user account has access permissions to view. Browse through the contents of the tree, or right-click an entry, and select **Search** from the pop-up menu.

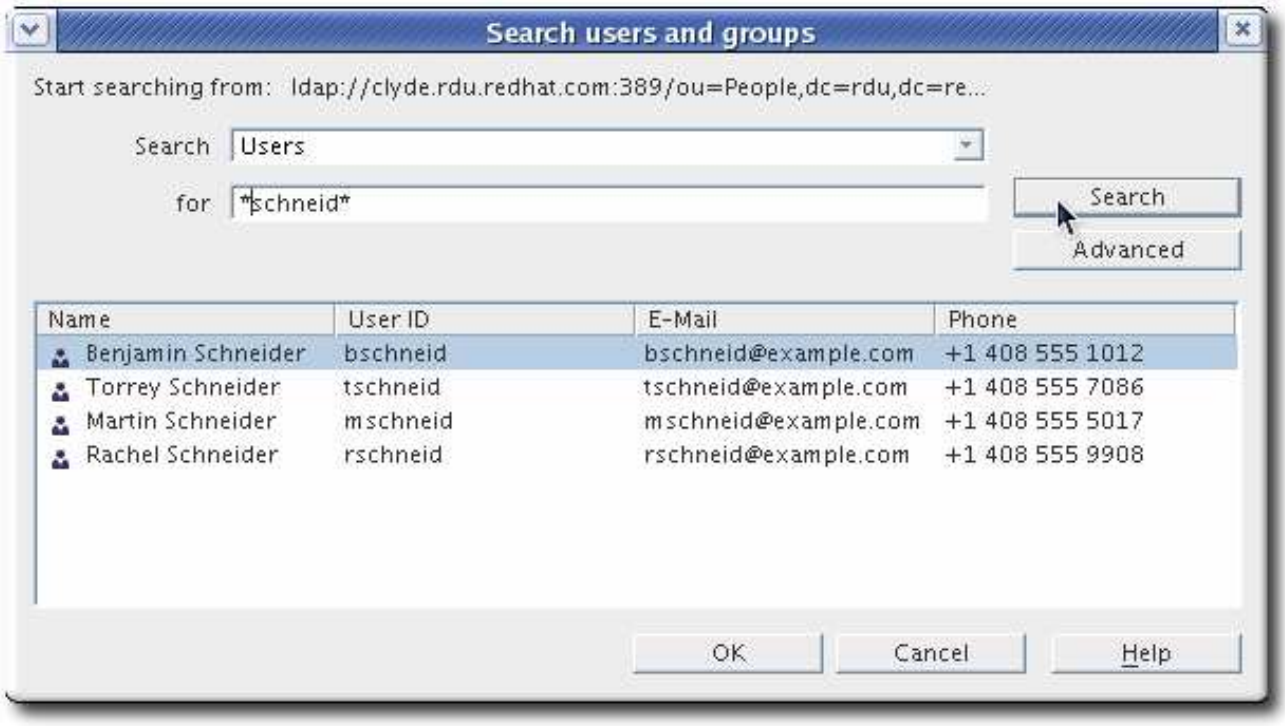


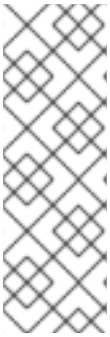
Figure 14.2. Searching for Entries

WARNING

Do not modify the contents of the **o=NetscapeRoot** suffix using the **Directory** tab unless instructed to do so by Red Hat technical support.

14.3. USING LDAPSEARCH

The **ldapsearch** command-line utility can locate and retrieve directory entries. This utility opens a connection to the specified server using the specified identity and credentials and locates entries based on a specified search filter. The search scope can include a single entry (**-s base**), an entry's immediate subentries (**-s one**), or an entire tree or subtree (**-s sub**).



NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=bjensen,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless **dc:example** has explicitly been added as an attribute in that entry.

Search results are returned in LDIF format. LDIF is defined in RFC 2849 and is described in detail in Appendix B, *LDAP Data Interchange Format*.

This section contains information about the following topics:

- Section 14.3.1, "ldapsearch Command-Line Format"
- Section 14.3.2, "Commonly Used ldapsearch Options"
- Section 14.3.3, "Using Special Characters"

14.3.1. ldapsearch Command-Line Format

The **ldapsearch** command must use the following format:

```
# ldapsearch [-x | -Y mechanism] [options] [search_filter] [list_of_attributes]
```

- Either **-x** (to use simple binds) or **-Y** (to set the SASL mechanism) must be used to configure the type of connection.
- *options* is a series of command-line options. These must be specified before the search filter, if any are used.
- *search_filter* is an LDAP search filter as described in Section 14.4, "LDAP Search Filters". Do not specify a separate search filter if search filters are specified in a file using the **-f** option.
- *list_of_attributes* is a list of attributes separated by a space. Specifying a list of attributes reduces the number of attributes returned in the search results. This list of attributes must appear after the search filter. For an example, see Section 14.5.6, "Displaying Subsets of Attributes". If a list of attributes is not specified, the search returns values for all attributes permitted by the access control set in the directory, with the exception of operational attributes.

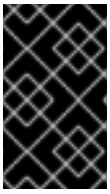
  For operational attributes to be returned as a result of a search operation, hey must be explicitly specified in the search command. To return all operational attributes of an object specify **+**. To retrieve regular attributes in addition to explicitly specified operational attributes, use an asterisk (*) in the list of attributes in the **ldapsearch** command.

  To retrieve only a list of matching DNs, use the special attribute **1.1**. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com \
    -b "dc=example,dc=com" -x "(objectclass=inetorgperson)" 1.1
```

14.3.2. Commonly Used ldapsearch Options

The following table lists the most commonly used **ldapsearch** command-line options. If a specified value contains a space ( ), the value should be surrounded by single or double quotation marks, such as **-b "cn=My Special Group,ou=groups,dc=example,dc=com"**.



IMPORTANT

The **ldapsearch** utility from OpenLDAP uses SASL connections by default. To perform a simple bind or to use TLS, use the **-x** argument to disable SASL and allow other connection methods.

Option	Description
-b	Specifies the starting point for the search. The value specified here must be a distinguished name that currently exists in the database. This is optional if the **LDAP_BASEDN** environment variable has been set to a base DN. The value specified in this option should be provided in single or double quotation marks. For example: `-b "cn=Barbara Jensen,ou=Product Development,dc=example,dc=com"` To search the root DSE entry, specify an empty string here, such as **-b ""** .
-D	Specifies the distinguished name with which to authenticate to the server. This is optional if anonymous access is supported by the server. If specified, this value must be a DN recognized by the Directory Server, and it must also have the authority to search for the entries. For example, **-D "uid=bjensen,dc=example,dc=com"**.

-H	Specifies an LDAP URL to use to connect to the server. For a traditional LDAP URL, this has the following format:
	`ldap[s]://hostname[:port]`
	The *port* is optional; it will use the default LDAP port of 389 or LDAPS port of 636 if the port is not given.
	This can also use an LDAPI URL, with each element separated by the HTML hex code **%2F**, rather than a forward slash (/):
	`ldapi://%2Ffull%2Fpath%2Fto%2Fslapd-example.socket`
	For LDAPI, specify the full path and filename of the LDAPI socket the server is listening to. Since this value is interpreted as an LDAP URL, the forward slash characters (/) in the path and filename must be escaped encoded as the URL escape value **%2F**.
	The **-H** option is used instead of**-h** and **-p**.

Option	Description
-h	Specifies the host name or IP address of the machine on which the Directory Server is installed. For example, **-h server.example.com**. If a host is not specified, **ldapsearch** uses the localhost.  NOTE Directory Server supports both IPv4 and IPv6 addresses.
-l	Specifies the maximum number of seconds to wait for a search request to complete. For example, **-l 300**. The default value for the ***nsslapd-timelimit*** attribute is **3600** seconds. Regardless of the value specified, **ldapsearch** will never wait longer than is allowed by the server's ***nsslapd-timelimit*** attribute.
-p	Specifies the TCP port number that the Directory Server uses. For example, **-p 1049**. The default is **389**. If **-h** is specified, **-p** must also be specified, even if it gives the default value.
-s *scope*	Specifies the scope of the search. The scope can be one of the following: **base** searches only the entry specified in the **-b** option or defined by the **LDAP_BASEDN** environment variable. **one** searches only the immediate children of the entry specified in the **-b** option. Only the children are searched; the actual entry specified in the **-b** option is not searched. **sub** searches the entry specified in the **-b** option and all of its descendants; that is, perform a subtree search starting at the point identified in the **-b** option. This is the default.
-W	Prompt for the password. If this option is not set, anonymous access is used. Alternatively, use the **-w** option to pass the password to the utility. Note that the password can be visible in the process list for other users and is saved in the shell's history.
-x	Disables the default SASL connection to allow simple binds.
-Y *SASL_mechanism*	Sets the SASL mechanism to use for authentication. If no mechanism is set, **ldapsearch** selects the best mechanism supported by the server. If If **-x** is not used, then the **-Y** option must be used.
-z *number*	Sets the maximum number of entries to return in a response to a search request. This value overwrites the server-side ***nsslapd-sizelimit*** parameter when binding using the root DN. wibrown>

14.3.3. Using Special Characters

When using the **ldapsearch** command-line utility, it may be necessary to specify values that contain characters that have special meaning to the command-line interpreter, such as space ( ), asterisk (*), or backslash (\). Enclose the value which has the special character in quotation marks (""). For example:

```
-D "cn=Barbara Jensen,ou=Product Development,dc=example,dc=com"
```

Depending on the command-line interpreter, use either single or double quotation marks. In general, use single quotation marks (') to enclose values. Use double quotation marks (") to allow variable interpolation if there are shell variables. Refer to the operating system documentation for more information.

14.4. LDAP SEARCH FILTERS

Search filters select the entries to be returned for a search operation. They are most commonly used with the **ldapsearch** command-line utility. When using **ldapsearch**, there can be multiple search filters in a file, with each filter on a separate line in the file, or a search filter can be specified directly on the command line.

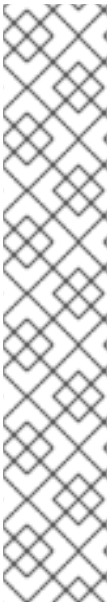
The basic syntax of a search filter is:

```
attribute operator value
```

For example:

```
buildingname>=alpha
```

In this example, ***buildingname*** is the attribute, **>=** is the operator, and **alpha** is the value. Filters can also be defined that use different attributes combined together with Boolean operators.



NOTE

When performing a substring search using a matching rule filter, use the asterisk (*) character as a wildcard to represent zero or more characters.

For example, to search for an attribute value that starts with the letter **l** and ends with the letter **n**, enter a **l*n** in the value portion of the search filter. Similarly, to search for all attribute values beginning with the letter **u**, enter a value of **u*** in the value portion of the search filter.

To search for a value that contains the asterisk (*) character, the asterisk must be escaped with the designated escape sequence, **5c2a**. For example, to search for all employees with ***businessCategory*** attribute values of **Example*Net product line**, enter the following value in the search filter:

```
Example\5c2a*Net product line
```

NOTE

A common mistake is to assume that the directory is searched based on the attributes used in the distinguished name. The distinguished name is only a unique identifier for the directory entry and cannot be used as a search key. Instead, search for entries based on the attribute-data pairs stored on the entry itself. Thus, if the distinguished name of an entry is **uid=bjensen,ou=People,dc=example,dc=com**, then a search for **dc=example** does not match that entry unless **dc:example** has explicitly been added as an attribute in that entry.

14.4.1. Using Attributes in Search Filters

The most basic sort of search looks for the presence of attributes or specific values in entries. There are many variations on *how* to look for attributes in entries. It is possible to check that the attribute merely exists, to match an exact value, or to list matches against a partial value.

A *presence* search uses a wild card (an asterisk) to return every entry which has that attribute set, regardless of value. For example, this returns every entry which has a ***manager*** attribute:

```
"(manager=*)"
```

It is also possible to search for an attribute with a specific value; this is called an *equality* search. For example:

```
"(cn=babs jensen)"
```

This search filter returns all entries that contain the common name Babs Jensen. Most of the time, equality searches are not case sensitive.

When an attribute has values associated with a language tag, all of the values are returned. Thus, the following two attribute values both match the **"(cn=babs jensen)"** filter:

```
cn: babs jensen
cn;lang-fr: babs jensen
```

It is also possible to search for a partial match on an attribute value, a *substring* index. For example:

```
"(description=*X.500*)"
"(sn=*nderson)"
"(givenname=car*)"
```

The length of the substring searches is configured in the substring index itself, as described in Section 13.6, “Changing the Width for Indexed Substring Searches” .

14.4.2. Using Operators in Search Filters

Operators in search filters set the relationship between the attribute and the given search value. For people searches, operators can be used to set a range, to return a last names within a subset of letters in the alphabet or employee numbers that come after a certain number.

```
"(employeeNumber>=500)"
"(sn~=suret)"
"(salary<=150000)"
```

Operators also enable phonetic and approximate searches, which allow more effective searches with imperfect information and are particularly useful in internationalized directories.

The operators that can be used in search filters are listed in Table 14.1, "Search Filter Operators". In addition to these search filters, special filters can be specified to work with a preferred language collation order. For information on how to search a directory with international charactersets, see Section D.4, "Searching an Internationalized Directory".

Table 14.1. Search Filter Operators

Search Type	Operator	Description
Equality	=	Returns entries containing attribute values that exactly match the specified value. For example, **cn=Bob Johnson**
Substring	=*string* string*	Returns entries containing attributes containing the specified substring. For example, **cn=Bob* cn=*Johnson cn=*John* cn=B*John**. The asterisk (*) indicates zero (0) or more characters.
Greater than or equal to	>=	Returns entries containing attributes that are greater than or equal to the specified value. For example, **buildingname >= alpha**.
Less than or equal to	<=	Returns entries containing attributes that are less than or equal to the specified value. For example, **buildingname <= alpha**.
Presence	=*	Returns entries containing one or more values for the specified attribute. For example, **cn=* telephoneNumber=* manager=***.
Approximate	~=	Returns entries containing the specified attribute with a value that is approximately equal to the value specified in the search filter. For example, **cn~=suret l~=san fransico** could return **cn=sarette l=san francisco**.

14.4.3. Using Compound Search Filters

Multiple search filter components can be combined using Boolean operators expressed in prefix notation as follows:

```
(Boolean-operator(filter)(filter)(filter)...)
```

Boolean-operator is any one of the Boolean operators listed in Table 14.2, "Search Filter Boolean Operators".

FOr example, this filter returns all entries that do not contain the specified value:

```
(!(cn=Ray Kultgen))
(!(objectClass=person))
```

Obviously, compound search filters are most useful when they are nested together into completed expressions:

```
(Boolean-operator(filter)((Boolean-operator(filter)(filter)))
```

These compound filters can be combined with other types of searches (approximate, substring, other operators) to get very detailed results. For example, this filter returns all entries whose organizational unit is **Marketing** and whose description field does not contain the substring **X.500**:

```
(&(ou=Marketing)(!(description=*X.500*)))
```

That filter can be expanded to return entries whose organizational unit is **Marketing**, that do not have the substring **X.500**, and that have Julie Fulmer or Cindy Zwaska as a manager:

```
(&(ou=Marketing)(!(description=*X.500*))(|(manager=cn=Julie
Fulmer,ou=Marketing,dc=example,dc=com)(manager=cn=Cindy
Zwaska,ou=Marketing,dc=example,dc=com)))
```

This filter returns all entries that do not represent a person and whose common name is similar to **printer3b**:

```
(&(!(objectClass=person))(cn~=printer3b))
```

Table 14.2. Search Filter Boolean Operators

Operator	Symbol	Description
AND	&	All specified filters must be true for the statement to be true. For example, *(&(filter)(filter)(filter)...)*.
OR	|	At least one specified filter must be true for the statement to be true. For example, *(|(filter)(filter)(filter)...)*
NOT	!	The specified statement must not be true for the statement to be true. Only one filter is affected by the **NOT** operator. For example,*(!(filter))*.

Boolean expressions are evaluated in the following order:

- Innermost to outermost parenthetical expressions first.
- All expressions from left to right.

14.4.4. Using Matching Rules

A *matching rule* tells the Directory Server how to compare two values (the value stored in the attribute and the value in the search filter). A matching rule also defines how to generate index keys. Matching rules are somewhat related to attribute syntaxes. Syntaxes define *the format* of an attribute value;

matching rules define how that format is compared and indexed.

There are three different types of matching rules:

- **EQUALITY** specifies how to compare two values for an equal match. For example, how to handle strings like "Fred" and "FRED". Search filters that test for equality (for example, *attribute=value*) use the EQUALITY rule. Equality (eq) indexes use the EQUALITY rule to generate the index keys. Update operations use the EQUALITY rule to compare values to be updated with values already in an entry.

- **ORDERING** specifies how to compare two values to see if one value is greater or less than another value. Search filters that set a range (for example, *attribute<=value* or *attribute>=value*) use the ORDERING rule. An index for an attribute with an ORDERING rule orders the equality values.

- **SUBSTR** specifies how to do substring matching. Substring search filters (for example, *attribute=*partial_string** or *attribute=*end_string*) use the SUBSTR rule. Substring (sub) indexes use the SUBSTR rule to generate the index keys.

IMPORTANT

A matching rule is required in order to support searching or indexing for the corresponding search filter or index type. For example, an attribute must have an EQUALITY matching rule in order to support equality search filters and eq indexes for that attribute. An attribute must have both an ORDERING matching rule and an EQUALITY matching rule in order to support range search filters and indexed range searches.

A search operation will be rejected with PROTOCOL_ERROR or UNWILLING_TO_PERFORM if an attempt is made to use a search filter for an attribute that has no corresponding matching rule.

Example 14.1. Matching Rules and Custom Attributes

Example Corp. administrators create a custom attribute type called ***MyFirstName*** with IA5 String (7-bit ASCII) syntax and an EQUALITY matching rule of caseExactIA5Match. An entry with a ***MyFirstName*** value of **Fred** is returned in a search with a filter of **(MyFirstName=Fred)**, but it is not returned for filters like **(MyFirstName=FRED)** and **(MyFirstName=fred) Fred**, **FRED**, and **fred** are all valid IA5 String values, but they do not match using the caseExactIA5Match rule.

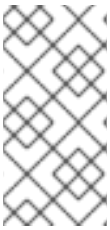
For all three variants of Fred to be returned in a search, then the ***MyFirstName*** should be defined to use the caseIgnoreIA5Match matching rule.

An extensible matching rule search filter can be used to search for an attribute value with a different matching rule than the one defined for the attribute. The matching rule must be compatible with the syntax of the attribute being searched. For example, to run a case insensitive search for an attribute that has a case-sensitive matching rule defined for it, specify a case insensitive matching rule in the search filter.

```
(MyFirstName:caseIgnoreIA5Match:=fred)
```

NOTE

Matching rules are used for searches in internationalized directories, to specify the language types to use for the results. This is covered in Section D.4, "Searching an Internationalized Directory".



NOTE

An index for an attributes uses whatever matching rules are defined for that attribute in its schema definition. Additional matching rules to use for an index can be configured using the ***nsMatchingRule*** attribute, as in Section 13.2.2, "Creating Indexes from the Command Line".

The syntax of the matching rule filter inserts a matching rule name or OID into the search filter:

attr:matchingRule:=value

- *attr* is an attribute belonging to entries being searched, such as ***cn*** or ***mail***.
- *matchingRule* is a string that contains the name or OID of the rule to use to match attribute values according to the required syntax.
- *value* is either the attribute value to search for or a relational operator plus the attribute value to search for. The syntax of the value of the filter depends on the matching rule format used.

A matching rule is actually a schema element, and, as with other schema elements is uniquely identified by an object identifier (OID).

Many of the matching rules defined for Red Hat Directory Server relate to language codes and set internationalized 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.

NOTE

Unlike other schema elements, additional matching rules cannot be added to the Directory Server configuration.

Most of the matching rules list in following list are used for equality indexes. Matching rules with *ordering* in their name are used for ordering indexes, and those with *substring* in their name are used for substring (SUBSTR) indexes. (The matching rules used for international matching and collation orders use a different naming scheme.)

Bitwise AND match

Performs bitwise **AND** matches.

OID: 1.2.840.113556.1.4.803

Compatible syntaxes: Typically used with **Integer** and numeric strings. Directory Server converts numeric strings automatically to integer.

Bitwise OR match

Performs bitwise **OR** matches.

OID: 1.2.840.113556.1.4.804

Compatible syntaxes: Typically used with **Integer** and numeric strings. Directory Server converts numeric strings automatically to integer.

booleanMatch

Evaluates whether the values to match are **TRUE** or **FALSE**

OID: 2.5.13.13

Compatible syntaxes: Boolean

caseExactIA5Match

Makes a case-sensitive comparison of values.

OID: 1.3.6.1.4.1.1466.109.114.1

Compatible syntaxes: **IA5** Syntax, URI

caseExactMatch

Makes a case-sensitive comparison of values.

OID: 2.5.13.5

Compatible syntaxes: Directory String, Printable String, OID

caseExactOrderingMatch

Allows case-sensitive ranged searches (less than and greater than).

OID: 2.5.13.6

Compatible syntaxes: Directory String, Printable String, OID

caseExactSubstringsMatch

Performs case-sensitive substring and index searches.

OID: 2.5.13.7

Compatible syntaxes: Directory String, Printable String, OID

caseIgnoreIA5Match

Performs case-insensitive comparisons of values.

OID: 1.3.6.1.4.1.1466.109.114.2

Compatible syntaxes: **IA5** Syntax, URI

caseIgnoreIA5SubstringsMatch

Performs case-insensitive searches on substrings and indexes.

OID: 1.3.6.1.4.1.1466.109.114.3

Compatible syntaxes: **IA5** Syntax, URI

caseIgnoreListMatch

Performs case-insensitive comparisons of values.

OID: 2.5.13.11

Compatible syntaxes: Postal address

caseIgnoreListSubstringsMatch

Performs case-insensitive searches on substrings and indexes.

OID: 2.5.13.12

Compatible syntaxes: Postal address

caseIgnoreMatch

Performs case-insensitive comparisons of values.

OID: 2.5.13.2

Compatible syntaxes: Directory String, Printable String, OID

caseIgnoreOrderingMatch

Allows case-insensitive ranged searches (less than and greater than).

OID: 2.5.13.3

Compatible syntaxes: Directory String, Printable String, OID

caseIgnoreSubstringsMatch

Performs case-insensitive searches on substrings and indexes.

OID: 2.5.13.4

Compatible syntaxes: Directory String, Printable String, OID

distinguishedNameMatch

Compares distinguished name values.

OID: 2.5.13.1

Compatible syntaxes: Distinguished name (DN)

generalizedTimeMatch

Compares values that are in a Generalized Time format.

OID: 2.5.13.27

Compatible syntaxes: Generalized Time

generalizedTimeOrderingMatch

Allows ranged searches (less than and greater than) on values that are in a Generalized Time format.

OID: 2.5.13.28

Compatible syntaxes: Generalized Time

integerMatch

Evaluates integer values.

OID: 2.5.13.14

Compatible syntaxes: Integer

integerOrderingMatch

Allows ranged searches (less than and greater than) on integer values.

OID: 2.5.13.15

Compatible syntaxes: Integer

keywordMatch

Compares the given search value to a string in an attribute value.

OID: 2.5.13.33

Compatible syntaxes: Directory String

numericStringMatch

Compares more general numeric values.

OID: 2.5.13.8

Compatible syntaxes: Numeric String

numericStringOrderingMatch

Allows ranged searches (less than and greater than) on more general numeric values.

OID: 2.5.13.9

Compatible syntaxes: Numeric String

numericStringSubstringMatch

Compares more general numeric values.

OID: 2.5.13.10

Compatible syntaxes: Numeric String

objectIdentifierMatch

Compares object identifier (OID) values.

OID: 2.5.13.0

Compatible syntaxes: OID

octetStringMatch

Evaluates octet string values.

OID: 2.5.13.17

Compatible syntaxes: Octet String

octetStringOrderingMatch

Supports ranged searches (less than and greater than) on a series of octet string values.

OID: 2.5.13.18

Compatible syntaxes: Octet String

telephoneNumberMatch

Evaluates telephone number values.

OID: 2.5.13.20

Compatible syntaxes: Telephone Number

telephoneNumberSubstringsMatch

Performs substring and index searches on telephone number values.

OID: 2.5.13.21

Compatible syntaxes: Telephone Number

uniqueMemberMatch

Compares both name and UID values.

OID: 2.5.13.23

Compatible syntaxes: Name and Optional UID

wordMatch

Compares the given search value to a string in an attribute value. This matching rule is case-insensitive.

OID: 2.5.13.32

Compatible syntaxes: Directory String

Table 14.3. Language Ordering Matching Rules

Matching Rule	Object Identifiers (OIDs)
English (Case Exact Ordering Match)	2.16.840.1.113730.3.3.2.11.3
Albanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.44.1

Matching Rule	Object Identifiers (OIDs)
Arabic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.1.1
Belorussian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.2.1
Bulgarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.3.1
Catalan (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.4.1
Chinese - Simplified (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.49.1
Chinese - Traditional (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.50.1
Croatian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.22.1
Czech (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.5.1
Danish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.6.1
Dutch (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.33.1
Dutch - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.34.1
English - US (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.11.1
English - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.12.1
English - Irish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.14.1
Estonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.16.1
Finnish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.17.1
French (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.18.1
French - Belgian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.19.1
French - Canadian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.20.1
French - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.21.1
German (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.7.1

Matching Rule	Object Identifiers (OIDs)
German - Austrian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.8.1
German - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.9.1
Greek (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.10.1
Hebrew (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.27.1
Hungarian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.23.1
Icelandic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.24.1
Italian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.25.1
Italian - Swiss (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.26.1
Japanese (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.28.1
Korean (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.29.1
Latvian, Lettish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.31.1
Lithuanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.30.1
Macedonian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.32.1
Norwegian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.35.1
Norwegian - Bokmul (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.36.1
Norwegian - Nynorsk (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.37.1
Polish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.38.1
Romanian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.39.1
Russian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.40.1
Serbian - Cyrillic (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.45.1
Serbian - Latin (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.41.1
Slovak (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.42.1

Matching Rule	Object Identifiers (OIDs)
Slovenian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.43.1
Spanish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.15.1
Swedish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.46.1
Turkish (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.47.1
Ukrainian (Case Insensitive Ordering Match)	2.16.840.1.113730.3.3.2.48.1

Table 14.4. Language Substring Matching Rules

Matching Rule	Object Identifiers (OIDs)
English (Case Exact Substring Match)	2.16.840.1.113730.3.3.2.11.3.6
Albanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.44.1.6
Arabic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.1.1.6
Belorussian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.2.1.6
Bulgarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.3.1.6
Catalan (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.4.1.6
Chinese - Simplified (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.49.1.6
Chinese - Traditional (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.50.1.6
Croatian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.22.1.6
Czech (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.5.1.6
Danish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.6.1.6
Dutch (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.33.1.6
Dutch - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.34.1.6
English - US (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.11.1.6

Matching Rule	Object Identifiers (OIDs)
English - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.12.1.6
English - Irish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.14.1.6
Estonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.16.1.6
Finnish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.17.1.6
French (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.18.1.6
French - Belgian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.19.1.6
French - Canadian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.20.1.6
French - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.21.1.6
German (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.7.1.6
German - Austrian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.8.1.6
German - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.9.1.6
Greek (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.10.1.6
Hebrew (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.27.1.6
Hungarian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.23.1.6
Icelandic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.24.1.6
Italian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.25.1.6
Italian - Swiss (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.26.1.6
Japanese (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.28.1.6
Korean (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.29.1.6
Latvian, Lettish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.31.1.6
Lithuanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.30.1.6

Matching Rule	Object Identifiers (OIDs)
Macedonian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.32.1.6
Norwegian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.35.1.6
Norwegian - Bokmul (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.36.1.6
Norwegian - Nynorsk (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.37.1.6
Polish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.38.1.6
Romanian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.39.1.6
Russian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.40.1.6
Serbian - Cyrillic (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.45.1.6
Serbian - Latin (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.41.1.6
Slovak (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.42.1.6
Slovenian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.43.1.6
Spanish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.15.1.6
Swedish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.46.1.6
Turkish (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.47.1.6
Ukrainian (Case Insensitive Substring Match)	2.16.840.1.113730.3.3.2.48.1.6

14.5. EXAMPLES OF COMMON LDAPSEARCHES

The next set of examples assumes the following:

- The search is for all entries in the directory.
- The directory is configured to support anonymous access for search and read. This means that no bind information has to be supplied in order to perform the search. For more information on anonymous access, see Section 18.13.1.1.3, “Granting Anonymous Access”.
- The server is located on a host named **server.example.com**.
- The server uses port number **389**. Since this is the default port, the port number does not have to be sent in the search request.

- TLS is enabled for the server on port **636** (the default LDAPS port number).
- The suffix under which all data are stored is **dc=example,dc=com**.

14.5.1. Returning All Entries

Given the previous information, the following call will return all entries in the directory (subject to the configured size and time resource limits):

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)"
```

"objectclass=*" is a search filter that matches any entry in the directory. Since every entry must have an object class, and the ***objectclass*** attribute is always indexed, this is a useful search filter to return every entry.

14.5.2. Specifying Search Filters on the Command Line

A search filter can be specified directly on the command line as long as the filter is enclosed in quotation marks ("filter"). If the filter is supplied with the command, do not specify the **-f** option. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "cn=babs jensen"
```

14.5.3. Searching the Root DSE Entry

The root DSE is a special entry that contains information about the directory server instance, including all of the suffixes supported by the local Directory Server. This entry can be searched by supplying a search base of "", a search scope of **base**, and a filter of **"objectclass=*"**. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -b "" -s base
"objectclass=*"
```

14.5.4. Searching the Schema Entry

The **cn=schema** entry is a special entry that contains information about the directory schema, such as object classes and attribute types.

The following command lists the content of the **cn=schema** entry:

```
# ldapsearch -o ldif-wrap=no -D "cn=Directory Manager" -W -b "cn=schema" \
   '(objectClass=subSchema)' -s sub objectClasses attributeTypes matchingRules \
   matchingRuleUse  dITStructureRules nameForms ITContentRules ldapSyntaxes
```

14.5.5. Using LDAP_BASEDN

To make searching easier, it is possible to set the search base using the **LDAP_BASEDN** environment variable. Doing this means that the search base does not have to be set with the **-b** option. For information on how to set environment variables, see the documentation for the operating system.

Typically, set **LDAP_BASEDN** to the directory's suffix value. Since the directory suffix is equal to the root, or topmost, entry in the directory, this causes all searches to begin from the directory's root entry.

For example, set **LDAP_BASEDN** to **dc=example,dc=com** and search for **cn=babs jensen** in the directory, use the following command-line call:

```
# export LDAP_BASEDN="dc=example,dc=com"

# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x "cn=babs jensen"
```

In this example, the default scope of **sub** is used because the **-s** option was not used to specify the scope.

14.5.6. Displaying Subsets of Attributes

The **ldapsearch** command returns all search results in LDIF format. By default, **ldapsearch** returns the entry's distinguished name and all of the attributes that a user is allowed to read. The directory access control can be set such that users are allowed to read only a subset of the attributes on any given directory entry. Only operational attributes are not returned. For operational attributes to be returned as a result of a search operation, explicitly specify them in the search command or use ***+*** to return all operational attributes.

It may not be necessary to have all of the attributes for an entry returned in the search results. The returned attributes can be limited to just a few specific attributes by specifying the required ones on the command line immediately after the search filter. For example, to show the ***cn*** and ***sn*** attributes for every entry in the directory, use the following command-line call:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" sn cn
```

14.5.7. Searching for Operational Attributes

Operational attributes are special attributes set by the Directory Server itself that are used by the server to perform maintenance tasks, like processing access control instructions. They also show specific information about the entry, like the time it was initially created and the name of the user who created it. Operational attributes are available for use on every entry in the directory, regardless of whether the attribute is specifically defined for the object class of the entry.

Operational attributes are not returned in regular **ldapsearch**es. According to RFC3673, use ***+*** to return all operational attributes in a search request:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" '+'
```

To return only some defined operational attributes, explicitly specify them in the **ldapsearch** request:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x "(objectclass=*)" creatorsName createTimestamp modifiersName modifyTimestamp
```

The complete list of operational attributes is in the "Operational Attributes and Object Classes" chapter in the *Red Hat Directory Server 10 Configuration, Command, and File Reference* .

NOTE

To return all of the regular entry attributes along with the specified operational attributes, use the special search attribute, **"*"**, in addition to the operational attributes that are listed.

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b
"dc=example,dc=com" -s sub -x "(objectclass=*)" "*" aci
```

The asterisk must be enclosed in quotation marks to prevent it from being interpreted by the shell.

14.5.8. Specifying Search Filters Using a File

Search filters can be entered into a file instead of entering them on the command line. In this case, specify each search filter on a separate line in the file. The **ldapsearch** command runs each search in the order in which it appears in the file.

For example:

```
sn=Francis
givenname=Richard
```

ldapsearch first finds all the entries with the surname **Francis**, then all the entries with the givenname **Richard**. If an entry is found that matches both search criteria, then the entry is returned twice.

For example, in this search, the filters are specified in a file named **searchdb**:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -f searchdb
```

The set of attributes returned here can be limited by specifying the attribute names at the end of the search line. For example, the following **ldapsearch** command performs both searches but returns only the DN and the ***givenname*** and ***sn*** attributes of each entry:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -f searchdb sn
givenname
```

14.5.9. Specifying DNs That Contain Commas in Search Filters

When a DN within a search filter contains a comma as part of its value, the comma must be escaped with a backslash (\). For example, to find everyone in the **example.com Bolivia, S.A.** subtree, use the following command:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -s base -b
"l=Bolivia\,S.A.,dc=example,dc=com" "objectclass=*"
```

14.5.10. Using a Client Certificate to Bind to Directory Server

See Section 9.8.4, “Authenticating Using a Certificate” .

14.5.11. Searching with Language Matching Rules

To explicitly submit a matching rule in a search filter, insert the matching rule after the attribute:

```
attr:matchingRule:=value
```

Matching rules are frequently used for searching internationalized directories. For example, this seearches for the department numbers after N4709 in the Swedish (**2.16.840.1.113730.3.3.2.46.1**) matching rule.

```
departmentNumber:2.16.840.1.113730.3.3.2.46.1:=>= N4709
```

More examples of performing internationalized searches are given in Section D.4, "Searching an Internationalized Directory".

14.5.12. Searching for Attributes with Bit Field Values

Bitwise searches use the bitwise AND or bitwise OR matching rules to perform bitwise search operaions on attributes with values that are bit fields.

NOTE

Attributes with values for bit fields are not common in LDAP. (No default Directory Server schema use bit fields as attribute syntax.) However, several LDAP syntaxes support integer-style values. Custom attributes can be defined which use bit field values, and applications can use those custom attributes to perform bitwise operations against bit field values.

The bitwise AND matching rule (**1.2.840.113556.1.4.803**) checks that the bit given in the assertion value is set in the bit field attribute value. (This is somewhat analogous to an equality search.) In this example, the userAccountControl value must be set to the bit representing 2.

```
"(UserAccountControl:1.2.840.113556.1.4.803:=2)"
```

In this example, the userAccountControl value must have all of the bits set that are set in the value 6 (bits 2 and 4).

```
"(UserAccountControl:1.2.840.113556.1.4.803:=6)"
```

The bitwise OR matching rule (**1.2.840.113556.1.4.804**) checks to see if *any* of the bits in the assertion string are represented in the attribute value. (This is somewhat analogous to a substring search.) In this example, the userAccountControl value must have any of the bits which are set in the bit field of 6, meaning that the attribute value can be 2, 4, or 6.

```
"(UserAccountControl:1.2.840.113556.1.4.804:=6)"
```

Bitwise searches can be used with Windows-Red Hat Enterprise Linux integration, such as using Samba file servers.

NOTE

Microsoft has good documentation on bitwise operators at http://msdn.microsoft.com/en-us/library/aa746475.

14.6. USING PERSISTENT SEARCH

A persistent search is an **ldapsearch** which remains open even after the initial search results are returned.

IMPORTANT

The OpenLDAP client tools with Red Hat Enterprise Linux do not support persistent searches. The server itself, however, does. Other LDAP clients must be used to perform persistent searches.

The purpose of a persistent search is to provide a continuous list of changes to the directory entries as well as the complete entries themselves, something like a hybrid search and changelog. Therefore, the search command must specify what entries to return (the search parameters) and what changes cause an entry to be returned (entry change parameters).

Persistent searches are especially useful for applications or clients which access the Directory Server and provide two important benefits:

- Keep a consistent and current local cache.

  Any client will query local cache before trying to connect to and query the directory. Persistent searches provide the local cache necessary to improve performance for these clients.

- Automatically initiate directory actions.

  The persistent cache can be automatically updated as entries are modified, and the persistent search results can display what kind of modification was performed on the entry. Another application can use that output to update entries automatically, such as automatically creating an email account on a mail server for new users or generating a unique user ID number.

There are some performance considerations when running persistent searches, as well:

- The **ldapsearch** does not send a notification when the client disconnects, and the change notifications are not sent for any changes made while the search is disconnected. This means that the client's cache will not be updated if it is ever disconnected and there is no good way to update the cache with any new, modified, or deleted entries that were changed while it was disconnected.

- An attacker could open a large number of persistent searches to launch a denial of service attack.

- A persistent search requires leaving open a TCP connection between the Directory Server and client. This should only be done if the server is configured to allow a lot of client connections and has a way to close idle connections.

In the access logs, a persistent search is identified with the tag **options=persistent**.

```
[12/Jan/2009:12:51:54.899423510 -0500] conn=19636710736396323 op=0 SRCH
base="dc=example,dc=com" scope=2 filter="(objectClass=person)" attrs=ALL options=persistent
```

14.7. SEARCHING WITH SPECIFIED CONTROLS

The Directory Server has defined controls in its ***supportedControls*** attribute in its DSE. Some of these define server operations like replication; other are allowed extended operations like get effective rights or dereferencing controls which clients can pass through LDAP operations to the server.

These controls can be specified using the **-E** option by giving the control OID, its criticality for the **ldapsearch**, and any information required for the control operation.

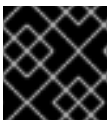
```
-E '[!]control_OID:control_information'
```

Some controls, like server-side sorting and simple paged results, have an alias that can be used to pass the control to the search operation. When the control alias is used, then the results are formatted, since the control is recognized by the client.

14.7.1. Retrieving Effective User Rights

A get effective-rights search control is passed using the control OID. For example:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x -E '!1.3.6.1.4.1.42.2.27.9.5.2=:dn:uid=jsmith,ou=people,dc=example,dc=com' "
(objectclass=*)"
```



IMPORTANT

When a control is passed with its OID, the results from the search are unformatted.

Get effective rights searches are covered in much more detail in the access control chapter, Section 18.14, “Checking Access Rights on Entries (Get Effective Rights)” .

14.7.2. Using Server-Side Sorting

Server-side sorting is performed as other control operations, using the **-E** flag and the **sss** control alias. The structure of the operation sets the attribute by which to sort the results and, optionally, the sort order and ordering rule.

```
-E sss=[-]attribute_name:[ordering_rule_OID]
```

The dash (**-**) is an optional flag that reverses the sort order, which naturally runs descending. The matching rule tables in Section 14.4.4, “Using Matching Rules” contain the ordering rules supported by the Directory Server.

For example:

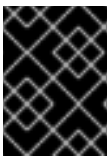
```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -b "dc=example,dc=com"
-s sub -x -E sss=-uidNumber:2.5.13.15 "(objectclass=*)"
```

14.7.3. Performing Dereferencing Searches

A *dereferencing* search is a quick way to track back over cross-references in an entry and return information about the referenced entry. For example, a group entry contains references to its member's user entries. A regular search first searches for the group, then lists its members, and then requires a

separate search for each member. A dereferencing search for the group entry returns information about the members – such as their locations, email addresses, or managers – *along with* the information for the group, all in a single search request.

Dereferencing simplifies many client operations and reduces the number of search operations that are performed. Cross-links show relationships between entries. Some operations may require getting a list of cross-links from one entry and then performing a series of subsequent searches to get information from each entry on the list. Dereferencing allows those sequences of searches to be consolidated into a single search.



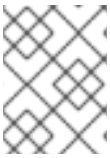
IMPORTANT

Dereferencing operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support dereferencing searches.

The format of the dereference arguments is:

```
-E 'deref=deref_attribute:list_of_attributes'
```

The *deref_attribute* is the attribute in the search target that contains the reference. This can be any attribute which has a DN for its value, such as **member** or **manager**.



NOTE

Not only must the value of the *deref_attribute* be a DN, but the actual defined syntax for the attribute must be DN syntax (**1.3.6.1.4.1.1466.115.121.1.12**).

The *list_of_attributes* is one or more attributes in the referenced entry which will be returned along with the primary search results. Multiple attributes can be separated by commas, like **l,mail,cn**.

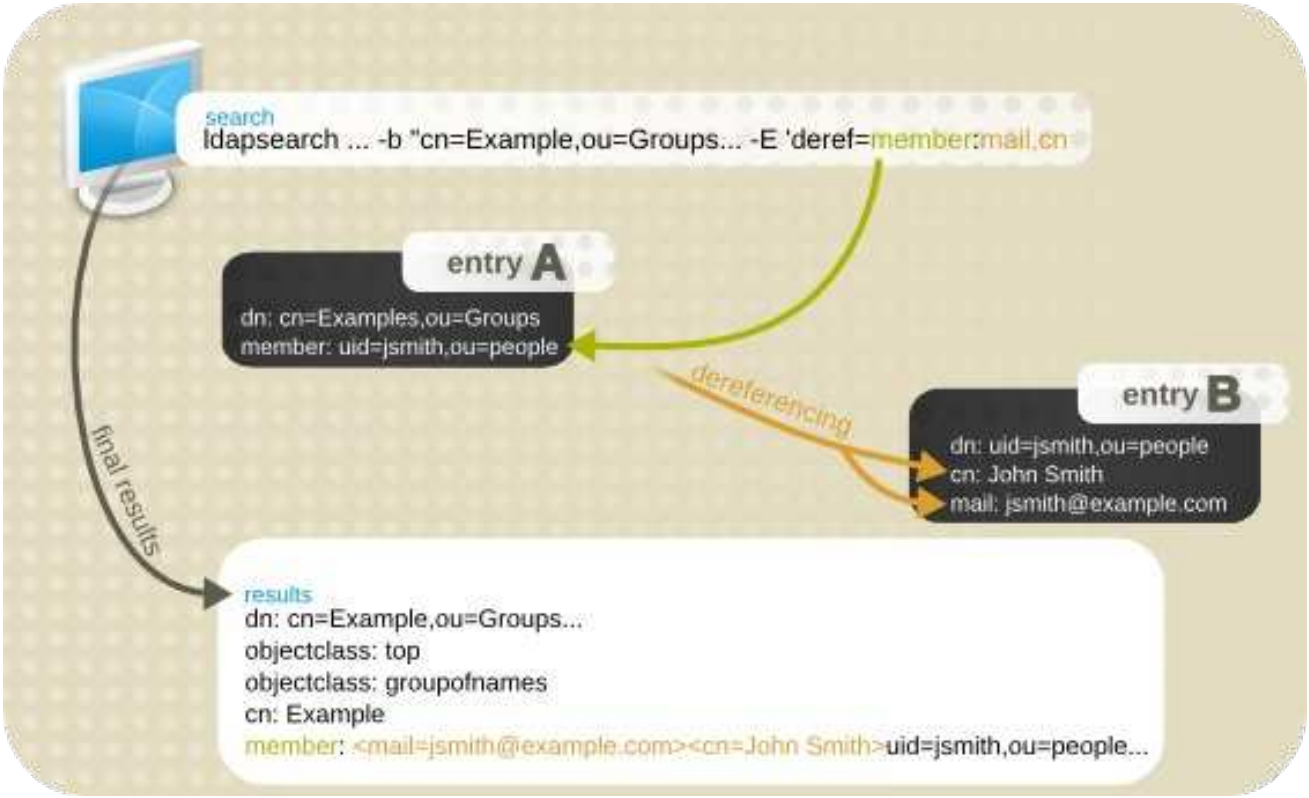


Figure 14.3. Simple Dereferencing Search Command

The requested dereferenced information requested in the search argument is returned with the rest of the search results. For example, this dereferencing search tells the server to use the ***member*** attribute in the search target entry (the Engineers group) as the *deref_attribute*. It then returns the locality attribute for each member.

```
# ldapsearch -x -D "cn=Directory Manager" -W -b "cn=Example,ou=Groups,dc=example,dc=com" -E
'deref=member:mail,cn' "(objectclass=*)"

# Engineers, Groups, example.com
dn: cn=Engineers,ou=Groups,dc=example,dc=com
control: 1.3.6.1.4.1.4203.666.5.16 false MIQAAADNMIQAAAA1BAZtZW1iZXIEK2NuPURld

mVsb3BlcnMsIG91PUdyb3VwcywgZGM9ZXhhbXBsZSxkYz1jb20whAAAADIEBm1lbWJlcgQoY249VG

VzdGVycywgb3U9R3JvdXBzLCBkYz1leGFtcGxlLGRjPWNvbTCEAAAAVAQGbWVtYmVyBCp1aWQ9ZW5

nLCBvdT1lbmdpbmVlcmluZywgZGM9ZXhhbXBsZSxkYz1jb22ghAAAABowhAAAABQEAWwxhAAAAAs
E
 CUNhbWJyaWRnZQ==
# member: <mail=jsmith@example.com><cn=John
Smith>;uid=jsmith,ou=people,dc=example,dc=com
 objectClass: top
objectClass: inetuser
objectClass: groupofnames
cn: Engineers
member: uid=jsmith,ou=people,dc=example,dc=com
```

14.7.4. Using Simple Paged Results

Search results can be very large, and a part of processing the results is organizing the results. One method of doing this is using *simple paged results*, a control that breaks the results into pages of a certain length.

The simple paged results control sets the number of entries to display at a time. The results can be scrolled through one page at a time which makes the results easier to digest. The full behavior of the control is described in RFC 2696.

Simple paged results are implemented as an LDAP control extension for the Directory Server. Its OID is **1.2.840.113556.1.4.319**.

How Simple Paged Results Work

When you start a simple paged results search:

1. The client sends the search to the server, together with the paged results control and with how many records to return in the first page.

2. Before Directory Server starts returning data, the server generates an estimate how many records can be returned in total.

   The estimate of records is not an exact number. The total number of records returned can be lower than the estimate. The reasons for such a scenario include

- attributes used in the search filter do not exist in the index. For an optimal result, all queried attributes must be indexed.
- before an entry is send to the client, access control lists (ACL) are validated. Insufficient permissions can prevent the entry from being returned.

After generating the estimate, the server sends the first set of results, a cookie, and the estimated number of records.

3. The returned records are displayed in the client. The user can now enter how many records should be returned in the next request. The requested number is now sent, together with the cookie, to the server.
4. The server retrieves the requested number of records from the database and sends them together with a cookie to the client.
5. The previous two steps are repeated until all records are sent or the search is cancelled.

Simple Paged Results and OpenLDAP Tools

The format of the simple paged result search option with **ldapsearch** is:

```
-E pg=size
```

The *size* value is the page size, or the number of entries to include per page. For example:

```
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 tag at the end shows the configured page size (the number in parentheses) from the search. After the colon, one enters the page size for the next page, so entering **5** as shown would open the next page of results with five entries.

IMPORTANT

Simple paged results operations must be done using OpenLDAP command-line tools version 2.4.18 or later or other clients which support simple paged results, such as Perl Net::LDAP.

Simple Paged Results and Server-Side Sorting

Simple paged results can be used together with server-side sorting. Server-side sorting is a control which performs the sort process on the server rather than in a client; this is usually done for a search which uses a particular matching rule. (This behavior is defined in RFC 2891.) The OpenLDAP client tools

do not support server-side sort with the simple paged results control, but other LDAP utilities such as Perl Net::LDAP do support both.

Multiple Simple Paged Results Requests on a Single Connection

Some clients may open a single connection to the Directory Server, but send multiple operation requests, including multiple search requests using the simple paged results extension.

Directory Server can manage and interpret multiple simple paged searches. Each search is added as an entry in an array. When the paged search request is first sent, there is a cookie created and associated with the search results. Each page of results is returned with that cookie, and that cookie is used to request the next page of results. On the last page, the cookie is empty, signalling the end of the results. This keeps each set of search results separate.

When there are multiple simple paged results on a single connection, the timeout limits are still observed, but *all* open search requests much reach their configured time limit before *any* paged search is disconnected.

Simple Paged Results, Contrasted with VLV Indexes

VLV indexes are similar to simple paged results in that they also return a usable browsing list of results. The main difference is in how that list is generated. Simple paged results are calculated per search, while VLV indexes are a permanent list. Overall, VLV indexes are faster for searches, but do require some server-side configuration and overhead for the server to maintain.

NOTE

Simple paged results and VLV indexes *cannot* be used on the same search. Simple paged results would attempt to manipulate the VLV index, which is already a browsing index. If the control is passed for a search using a VLV index, then the server returns an **UNWILLING_TO_PERFORM** error.

For more information on VLV indexes, see Section 13.4, “Creating Browsing (VLV) Indexes” .

14.7.5. Pre- and Post-read Entry Response Controls

Red Hat Directory Server supports pre- and post-read entry response controls according to RFC 4527. If a client requests one or both response controls, an LDAP search entry is returned, that contains the attribute's value before and after the update.

When the pre-read control is used, an LDAP search query is returned containing the specified attribute's value before modification. When the post-read control is used, the query contains the attribute's value after modification. Both controls can be used at the same time. For example, to update the ***description*** attribute and display the value before and after the modification:

```
# ldapmodify -D "cn=Directory Manager" -W -x \
    -e \!preread=description -e \!postread=description
dn: uid=user,ou=People,dc=example,dc=com
changetype: modify
replace: description
description: new description

modifying entry "uid=user,ou=People,dc=example,dc=com"
control: 1.3.6.1.1.13.1 false ZCkEJXVpZD1qdXNlcixvdT1QZW9wbGUsZGM9ZXhhbXBsZSxk
 Yz1jb20wAA==
# ==> preread
```

```
dn: uid=user,ou=People,dc=example,dc=com
description: old description
# <== preread
control: 1.3.6.1.1.13.2 false ZEsEJXVpZD1qdXNlcixvdT1QZW9wbGUsZGM9ZXhhbXBsZSxk
Yz1jb20wIjAgBAtkZXNjcmlwdGlvbjERBA9uZXcgZGVzY3JpcHRpb24=
# ==> postread
dn: uid=user,ou=People,dc=example,dc=com
description: new description
# <== postread
```

CHAPTER 15. MANAGING REPLICATION

Replication is the mechanism by which directory data is automatically synchronized from one Red Hat Directory Server instance to another; it is an important mechanism for extending the directory service beyond a single server configuration. This chapter describes the tasks to be performed on the master and consumer servers to set up single-master replication, multi-master replication, and cascading replication.

15.1. REPLICATION OVERVIEW

Replication is the mechanism by which directory data is automatically synchronized from one Directory Server to another. Updates of any kind — entry additions, modifications, or even deletions — are automatically mirrored to other Directory Servers using replication.

15.1.1. What Directory Units Are Replicated

The smallest unit of of the directory which can be replicated is a database. This means that one can replicate an entire database but not a subtree within a database. Therefore, when creating the directory tree, consider any replication plans as part of determining how to distribute information.

Replication also requires that one database correspond to one suffix. This means that a suffix (or namespace) that is distributed over two or more databases using custom distribution logic cannot be replicated. For more information on this topic, see Section 2.2, "Creating and Maintaining Databases".

15.1.2. Read-Write and Read-Only Replicas

A database that participates in replication is called a *replica*. There are two kinds of replicas: read-write or read-only. A *read-write replica* contains master copies of directory information and can be updated. A *read-only replica* services read, search, and compare requests, but refers all update operations to read-write replicas. A server can hold any number of read-only or read-write replicas.

15.1.3. Suppliers and Consumers

A server that holds a replica that is copied to a replica on a different server is called a *supplier* for that replica. A server that holds a replica that is copied from a different server is called a *consumer* for that replica. Generally, the replica on the supplier server is a read-write replica, and the one on the consumer server is a read-only replica, with two exceptions:

- In the case of cascading replication, the hub server holds a read-only replica that it supplies to consumers. Section 15.3.3, "Cascading Replication" has more information.
- In the case of multi-master replication, the *masters* are both suppliers and consumers for the same information. For more information, see Section 15.3.2, "Multi-Master Replication".

Replication is always initiated by the supplier server, never by the consumer (*supplier-initiated replication*). Supplier-initiated replication allows a supplier server to be configured to push data to multiple consumer servers.

15.1.4. Changelog

Every supplier server maintains a *changelog*, a record of all changes that a supplier or hub needs to send to its consumers. A changelog is a special kind of database that describes the modifications that have occurred on a replica. The supplier server then replays these modifications to the replicas stored on consumer servers or to other suppliers, in the case of multi-master replication.

When an entry is modified, a change record describing the LDAP operation that was performed is recorded in the changelog.

The changelog uses the same database environment as the main database. Implementing the changelog as part of the main database ensures the database and changelog are always synchronized, reduces the required database cache size, and simplifies backup and restore operations.

IMPORTANT

The changelog only write RUV entries to the database when the server is shut down, and otherwise the RUVs are managed in memory. When you back up the database of a master, use the **db2bak.pl** utility or the Directory Server Console. Both ways, the RUVs are written to the database before the backup starts.

In Directory Server, the changelog is only intended for internal use by the server.

15.1.5. Replication Identity

When replication occurs between two servers, the replication process uses a special entry, called the *replication manager* entry, to identify replication protocol exchanges and to control access to the directory data. The replication manager entry, or any entry used during replication, must meet the following criteria:

- It is created on the consumer server and *not* on the supplier server.
- Create this entry on *every* server that receives updates from another server, meaning on every hub or dedicated consumer.
- When a replica is configured as a consumer or hub, this entry must be specified as the one authorized to perform replication updates.
- The replication agreement is created on the supplier server, the DN of this entry must be specified in the replication agreement.
- This entry, with its special user profile, bypasses all access control rules defined on the consumer server for the database involved in that replication agreement.

NOTE

In the Directory Server Console, this replication manager entry is referred to as the *supplier bind DN*, which may be misleading because the entry does not actually exist on the supplier server. It is called the supplier bind DN because it is the entry which the supplier uses to bind to the consumer. This entry actually exists, then, on the consumer.

For more information on creating the replication manager entry, see Section 15.4, "Creating the Supplier Bind DN Entry".

15.1.6. Replication Agreement

Directory Servers use replication agreements to define their replication configuration. A replication agreement describes replication between *one* supplier and *one* consumer only. The agreement is configured on the supplier server and must specify all required replication information:

- The database to be replicated.
- The consumer server to which the data is pushed.
- The days and times during which replication can occur.
- The DN and credentials that the supplier server must use to bind (the replication manager entry or supplier bind DN).
- How the connection is secured (TLS, client authentication).
- Any attributes that will not be replicated (fractional replication).

15.1.7. Replicating a Subset of Attributes with Fractional Replication

Fractional replication sets a specific subset of attributes that will not be transmitted from a supplier to the consumer (or another supplier). Administrators can therefore replicate a database without replicating all the information that it contains or all of the information in every entry.

Fractional replication is enabled and configured per replication agreement, not per entry. Excluding attributes from replication is applied equally to all entries within the replication agreement's scope.

As far as the consumer server is concerned, the excluded attributes always have no value. Therefore, a client performing a search against the consumer server will never see the excluded attributes. Similarly, should it perform a search that specifies those attributes in its filter, no entries will match.

For attributes that are defined as optional (**MAY** keyword) in the schema, it is possible to set different attributes to be replicated for an incremental update and a total update. The incremental update list (***nsDS5ReplicatedAttributeList***) must always be set to enable fractional replication; if that is the only attribute set, then it applies to both incremental and total updates. The optional ***nsDS5ReplicatedAttributeListTotal*** attribute sets an additional fractional replication list for total updates. This is described in Section 15.10.1, "Setting Different Fractional Replication Attributes for Total and Incremental Updates".

NOTE

An update to an excluded attribute still triggers a modify event and generates an empty replication update. The ***nsds5ReplicaStripAttrs*** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attribtes like ***modifiersName***.

If a replication event is *not* empty, the stripped attributes *are* replicated. These attributes are removed from updates only if the event would otherwise be emtpy.

15.1.7.1. The Replication Keep-alive Entry

When you update an attribute on a master, the change sequence number (CSN) is increased on the master. In a replication topology, this server now connects to the first consumer and compares the local CSN with the CSN on the consumer. If it is lower, the update is retrieved from the local changelog and replicated to the consumer. In a replication topology with fractional replication enabled, this can cause problems: For example, if only attributes are updated on the master that are excluded from replication, no update to replicate is found, and therefore the CSN is not updated on the consumer. In certain scenarios, such as when only attributes are updated on a master that are excluded from replication, unnecessary searching for updates on the supplier can cause other servers to receive the data later than needed . To work around this problem, Directory Server uses keep-alive entries.

If all updated attributes on the master are excluded from replication and the number of skipped updates exceeds 100, the ***keepalivetimestamp*** attribute is updated on the supplier and replicated to the consumer. Because the ***keepalivetimestamp*** attribute is not excluded from replication, the update of the keep-alive entry is replicated, the CSN on the consumer is updated, and then equal to the one on the supplier. The next time the supplier connects to the consumer, only updates that are newer than the CSN on the consumer are searched. This reduces the amount of time spent by a supplier to search for new updates to send.

The replication keep-alive entry is created on demand on a master and contains the replica ID of the master in the distinguished name (DN). Each keep-alive entry is specific to a given master. For example:

```
dn: cn=repl keep alive 14,dc=example,dc=com
objectclass: top
objectclass: ldapsubentry
objectclass: extensibleObject
cn: repl keep alive 14
keepalivetimestamp: 20170227190346Z
```

The keep-alive entry is updated in the following situations (if it does not exist before the update, it is created first):

- When a fractional replication agreement skips more than 100 updates and does not send any updates before ending the replication session.
- When a master initializes a consumer, initially it creates its own keep-alive entry. A consumer that is also a master does not create its own keep-alive entry unless it also initializes another consumer.

15.2. CONFIGURING REPLICATION FROM THE COMMAND LINE

Replication can be configured on the command line by creating the appropriate replica and agreement entries on the servers. The process follows the same order as setting up replication through the Directory Server Console:

1. Create the supplier bind DN on every consumer, hub, and multi-master supplier (Section 15.4, “Creating the Supplier Bind DN Entry”).
2. If the corresponding database and suffix do not exist on one of the replicas, create it (Section 2.1.1, “Creating Suffixes”).
3. Configure the supplier replicas (Section 15.2.1, “Configuring Suppliers from the Command Line”).
4. Configure consumers (Section 15.2.2, “Configuring Consumers Using the Command Line”).
5. Configure hubs for cascading replication (Section 15.2.3, “Configuring Hubs from the Command Line”).

6. Create the replication agreements (Section 15.2.4, "Configuring Replication Agreements from the Command Line"). For cascading replication, create the agreement between the supplier and hub, then between the hub and consumers; for multi-master, create the agreements between all suppliers, then between the suppliers and consumers.

7. Lastly, initialize all of the consumers (Section 15.2.5, "Initializing Consumers Online from the Command Line"), if the consumers were not initialized when the replication agreement was created.

15.2.1. Configuring Suppliers from the Command Line

There are two steps to setting up the supplier replica. First, the changelog must be enabled, which allows the supplier to track changes to the Directory Server. Then, the supplier replica is created.

1. On the supplier server, use **ldapmodify** to create the changelog entry.

Example 15.1. Example Changelog Entry

```
# ldapmodify -D "cn=Directory Manager" -W -x -h supplier1.example.com -v -a

dn: cn=changelog5,cn=config
changetype: add
objectclass: top
objectclass: extensibleObject
cn: changelog5
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance/changelogdb
nsslapd-changelogmaxage: 10d
```

There are two important attributes with the changelog.

- ***nsslapd-changelogdir*** sets the directory where the changelog is kept.
- ***nsslapd-changelogmaxage*** sets how long the changelog is kept; since the changelog can get very large, this helps trim the changelog to prevent affecting server performance and using up disk space. If this parameter is not set, the default is for the changelog to be kept forever.

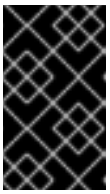
The changelog entry attributes are described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

2. Create the supplier replica.

Example 15.2. Example Supplier Replica Entry

```
# ldapmodify -D "cn=Directory Manager" -W -x -h supplier1.example.com -v -a

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 7
```

```
nsds5replicatype: 3
nsds5flags: 1
nsds5ReplicaPurgeDelay: 604800
nsds5ReplicaBindDN: cn=replication manager,cn=config
```



IMPORTANT

You must set the ***cn*** parameter of the replica entry to **replica** as shown in the example. Directory Server ignores the entry if you set the parameter to a different value.

The changelog entry attributes are described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

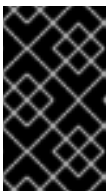
After creating every supplier which will take part in the replication setup, then begin creating the replication agreements.

15.2.2. Configuring Consumers Using the Command Line

To configure a consumer using the command line, the following settings are required on the consumer host:

1. Create the replica entry:

```
# ldapadd -D "cn=Directory Manager" -W -p 389 -h consumer.example.com -x

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 65535
nsds5replicatype: 2
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5flags: 0
```



IMPORTANT

You must set the ***cn*** parameter of the replica entry to **replica** as shown in the example. Directory Server ignores the entry if you set the parameter to a different value.

This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is.

2. Set the ***nsslapd-referral*** parameter to the LDAP URL of the supplier server and the ***nsslapd-state*** to **referral on update**. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h consumer.example.com -x
dn: cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
```

```
changetype: modify
replace: nsslapd-referral
nsslapd-referral: ldap://supplier.example.com:389/dc\=example\,dc\=com
-
replace: nsslapd-state
nsslapd-state: referral on update
```

For further details about the attributes used in the examples, see the corresponding sections in the *Red Hat Directory Server Configuration, Command, and File Reference* .

15.2.3. Configuring Hubs from the Command Line

Hubs are intermediate read-only replicas which receive updates from suppliers and pass them on to other consumers. These are part of the cascading replication scenario, described in Section 15.3.3, "Cascading Replication".Creating the hub has two steps: first, creating the changelog database since the hub keeps a record of changes sent by the supplier, and second, configuring the hub replica.

1. On the hub server, such as **hub1.example.com**, use **ldapmodify** to create the changelog entry.

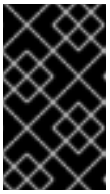
   ```
   # ldapmodify -D "cn=Directory Manager" -W -x -h hub1.example.com -v -a

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

   There is one important attribute with the changelog, ***nsslapd-changelogdir***, which sets the directory where the changelog is kept.

   The changelog entry attributes are described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

2. On the hub host, create the replica entry. This **ldapmodify** command creates a new hub replica on the **hub1.example.com** host for the **dc=example,dc=com** subtree.

   ```
   # ldapmodify -D "cn=Directory Manager" -W -x -h hub1.example.com -v -a

   dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
   changetype: add
   objectclass: top
   objectclass: nsds5replica
   objectclass: extensibleObject
   cn: replica
   nsds5replicaid: 65535
   nsds5replicaroot: dc=example,dc=com
   nsds5replicatype: 2
   nsds5ReplicaPurgeDelay: 604800
   nsds5ReplicaBindDN: cn=replication manager,cn=config
   nsds5flags: 1
   ```



IMPORTANT

You must set the ***cn*** parameter of the replica entry to **replica** as shown in the example. Directory Server ignores the entry if you set the parameter to a different value.

This entry identifies the database and suffix as participating in replication and sets what kind of replica the database is.

The changelog entry attributes are described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

15.2.4. Configuring Replication Agreements from the Command Line

When setting up replication agreements, first set them up between all suppliers, then between the suppliers and the hubs, and last between the hub and the consumers.

The replication agreement has to define eight separate attributes:

- The consumer host (***nsds5replicahost***) and port (***nsds5replicaport***).
- The DN for the supplier to use to bind with the consumer (***nsds5ReplicaBindDN***).
- The way that the supplier binds (***nsds5replicabindmethod***).
- Any credentials required (***nsDS5ReplicaCredentials***) for that bind method and specified DN.
- The subtree being replicated (***nsds5replicaroot***).
- The replication schedule (***nsds5replicaupdateschedule***).
- Any attributes which will *not* be replicated (***nsds5replicatedattributelist*** and ***nsDS5ReplicatedAttributeListTotal***).

Use **ldapmodify** to add a replication agreement to every supplier for every consumer which it will updated. For example:

Example 15.3. Example Replication Agreement Entry

```
dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
objectclass: top
objectclass: nsds5ReplicationAgreement
cn: ExampleAgreement
nsds5replicahost: consumer1
nsds5replicaport: 389
nsds5ReplicaBindDN: cn=replication manager,cn=config
nsds5replicabindmethod: SIMPLE
nsds5replicaroot: dc=example,dc=com
description: agreement between supplier1 and consumer1
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList
accountUnlockTime memberof
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
nsds5replicacredentials: secret
```

For descriptions of the parameters used in the example, and additional parameters you can set in the **cn=*agreement_name*,cn=replica,cn=*suffix_DN*,cn=mapping tree,cn=config** entry, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

After creating every replication agreement, begin initializing consumers.

15.2.4.1. Configuring Replication Partners to use Certificate-based Authentication

Instead of using a bind DN and password to authenticate to a replication partner, you can use certificate-based authentication.

The following procedure describes how to add a new server named **server2.example.com** to the replication topology and how to set up replication agreements between the new host and the existing **server1.example.com** using certificate-based authentication:

1. On both hosts, set up certificate-based authentication. For details, see Section 9.8.1, "Setting up Certificate-based Authentication".

2. On the **server1.example.com** host:

   a. Create accounts for both servers, such as **cn=server1,example,dc=com** and **cn=server2,dc=example,dc=com** and add the client certificates to the corresponding accounts. For details, see:

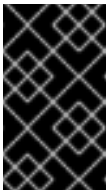
      - Section 3.1.3.1, "Adding an Entry Using **ldapadd**".
      - Section 9.8.2, "Adding a Certificate to a User"

      Both servers will later use these accounts and certificates to authenticate when they establish a replication connection to each other.

   b. Create a group, such as **cn=repl_server,ou=Groups,dc=example,dc=com**, and add both server accounts. See Section 8.1.3, "Creating Groups in the Command Line".

   c. Create the replica entry and set the ***nsds5ReplicaBindDNGroup*** attribute to the DN of the group created in the previous step:

```
# ldapmodify -D "cn=Directory Manager" -W -p 636 -h server1.example.com -x

dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: add
objectclass: top
objectclass: nsds5replica
objectclass: extensibleObject
cn: replica
nsds5replicaroot: dc=example,dc=com
nsds5replicaid: 7
nsds5replicatype: 3
nsds5flags: 1
nsds5ReplicaPurgeDelay: 604800
nsds5replicabinddngroup: cn=repl_server,ou=Groups,dc=example,dc=com
nsDS5ReplicaBindDNGroupCheckInterval: 0
```



IMPORTANT

You must set the ***cn*** parameter of the replica entry to **replica** as shown in the example. Directory Server ignores the entry if you set the parameter to a different value.

3. Initialize the new server:

   a. Create a temporary replication manager account, such as **cn=Replication Manager,cn=config**, on **server2.example.com**. See Section 15.4, "Creating the Supplier Bind DN Entry".

   b. On **server1.example.com**, create a temporary replication agreement which uses the account from the previous step for authentication:

```
# ldapmodify -D "cn=Directory Manager" -W -p 636 -h server1.example.com -x

dn: cn=temporary_agreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping
tree,cn=config
objectclass: top
objectclass: nsds5ReplicationAgreement
cn: temporary_agreement
nsds5replicahost: server2.example.com
nsds5replicaport: 636
nsds5replicabindmethod: SIMPLE
nsds5ReplicaBindDN: cn=Replication Manager,cn=config
nsds5replicacredentials: password_of_replication_manager_account
nsds5replicaroot: dc=example,dc=com
description: Temporary agreement between server1 and server2
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList
accountUnlockTime memberof
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
nsds5BeginReplicaRefresh: start
```

   This agreement uses the previously-created replication manager account to initialize the database. Before this initialization, the database on **server2.example.com** is empty and the accounts with the associated certificates do not exist. Therefore, replication using certificates is not possible before the database is initialized.

4. After the new server has been initialized:

   a. Remove the temporary replication agreement from **server1.example.com**:

```
# ldapdelete -D "cn=Directory Manager" -W -p 636 -h server1.example.com -x
"cn=temporary_agreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping
tree,cn=config"
```

   b. Remove the temporary replication manager account from **server2.example.com**:

```
# ldapdelete -D "cn=Directory Manager" -W -p 636 -h server2.example.com -x
"cn=Replication Manager,cn=config"
```

5. Create a replication agreement on both servers that use certificate-based authentication:

a. On **server1.example.com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 636 -h server1.example.com -x

dn: cn=example_agreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping
tree,cn=config
objectclass: top
objectclass: nsds5ReplicationAgreement
cn: example_agreement
nsds5replicahost: server2.example.com
nsds5replicaport: 636
nsds5replicabindmethod: SSLCLIENTAUTH
nsds5replicaroot: dc=example,dc=com
description: Agreement between server1 and server2
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList
accountUnlockTime memberof
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
nsDS5ReplicaTransportInfo: SSL
```

b. On **server2.example.com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 636 -h server2.example.com -x

dn: cn=example_agreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping
tree,cn=config
objectclass: top
objectclass: nsds5ReplicationAgreement
cn: example_agreement
nsds5replicahost: server1.example.com
nsds5replicaport: 636
nsds5replicabindmethod: SSLCLIENTAUTH
nsds5replicaroot: dc=example,dc=com
description: Agreement between server2 and server1
nsds5replicaupdateschedule: 0000-0500 1
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList
accountUnlockTime memberof
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
nsDS5ReplicaTransportInfo: SSL
```

6. To verify the replication works correctly, display the ***nsds5replicaLastUpdateStatus*** attribute in the replication agreement:

```
# ldapsearch -D "cn=Directory Manager" -W -p 636 -h server1.example.com -b
"cn=example_agreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config"
nsds5replicaLastUpdateStatus
```

For details about possible statuses, see the *Replication Agreement Status* appendix in the *Red Hat Directory Server Configuration, Command, and File Reference* .

15.2.5. Initializing Consumers Online from the Command Line

An online initialization can be initiated from the command line by adding the ***nsds5replicarefresh*** attribute to the replication agreement entry. If the attribute is included when the replication agreement

is created, initialization begins immediately. It can be added later to initialize the consumer at any time. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

```
# ldapsearch -x -h supplier1.example.com -p 389 -D "cn=Directory Manager" -W -s sub -b
cn=config "(objectclass=nsds5ReplicationAgreement)"
```

   This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the ***nsds5BeginReplicaRefresh*** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -x -h supplier1.example.com

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
replace: nsds5BeginReplicaRefresh
nsds5BeginReplicaRefresh: start
```

   ldapmodify does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the **ldapmodify** utility by hitting **Ctrl+C**.

When the initialization is complete, the ***nsds5BeginReplicaRefresh*** attribute is automatically deleted from the replication agreement entry.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Initializing consumers from the command line is also explained in Section 15.18.3, “Initializing Consumers Online Using the Command Line”. Manually initializing consumers is explained in Section 15.18.4, “Manual Consumer Initialization Using the Command Line”. The replication monitoring attributes are described in more detail in the *Red Hat Directory Server Configuration, Command, and File Reference*.

NOTE

For large databases, the ***nsslapd-idletimeout*** setting must be set to a large enough time period (or even an unlimited period) to allow the entire database to be initialized before the operation times out. Alternatively, the ***nsIdleTimeout*** setting for the supplier bind DN entry can be set high enough to allow the online initialization operation to complete, without having to change the global setting.

To keep data integrity, initialize the consumer databases from the appropriate supplier. Determining the correct supplier can be more difficult in mixed replication environments, but, even when manually initializing consumers, consider four things:

- Use one supplier, a *data master*, as the source for initializing consumers.

- Do not *reinitialize* a data master when the replication agreements are created. For example, do not initialize server1 from server2 if server2 has already been initialized from server1.
- For a multi-master scenario, initialize all of the other master servers in the configuration from one master.
- For cascading replication, initialize all of the hubs from a supplier, then initialize the consumers from the hubs.

15.3. REPLICATION SCENARIOS

- Section 15.3.1, "Single-Master Replication"
- Section 15.3.2, "Multi-Master Replication"
- Section 15.3.3, "Cascading Replication"

These basic strategies can be combined in a variety of ways to create the best replication environment.

NOTE

Whatever replication scenario is implemented, consider schema replication. To avoid conflict resolution loops, the Referential Integrity Plug-in should only be enabled on one supplier replica in a multi-master replication environment. The plug-in is off by default.

15.3.1. Single-Master Replication

In the simplest replication scenario, the master copy of directory data is held in a single read-write replica on one server called the *supplier server*. The supplier server also maintains changelog for this replica. On another server, called the *consumer server*, there can be multiple read-only replicas. Such scenarios are called *single-master configurations*. Figure 15.1, "Single-Master Replication" shows an example of single-master replication.

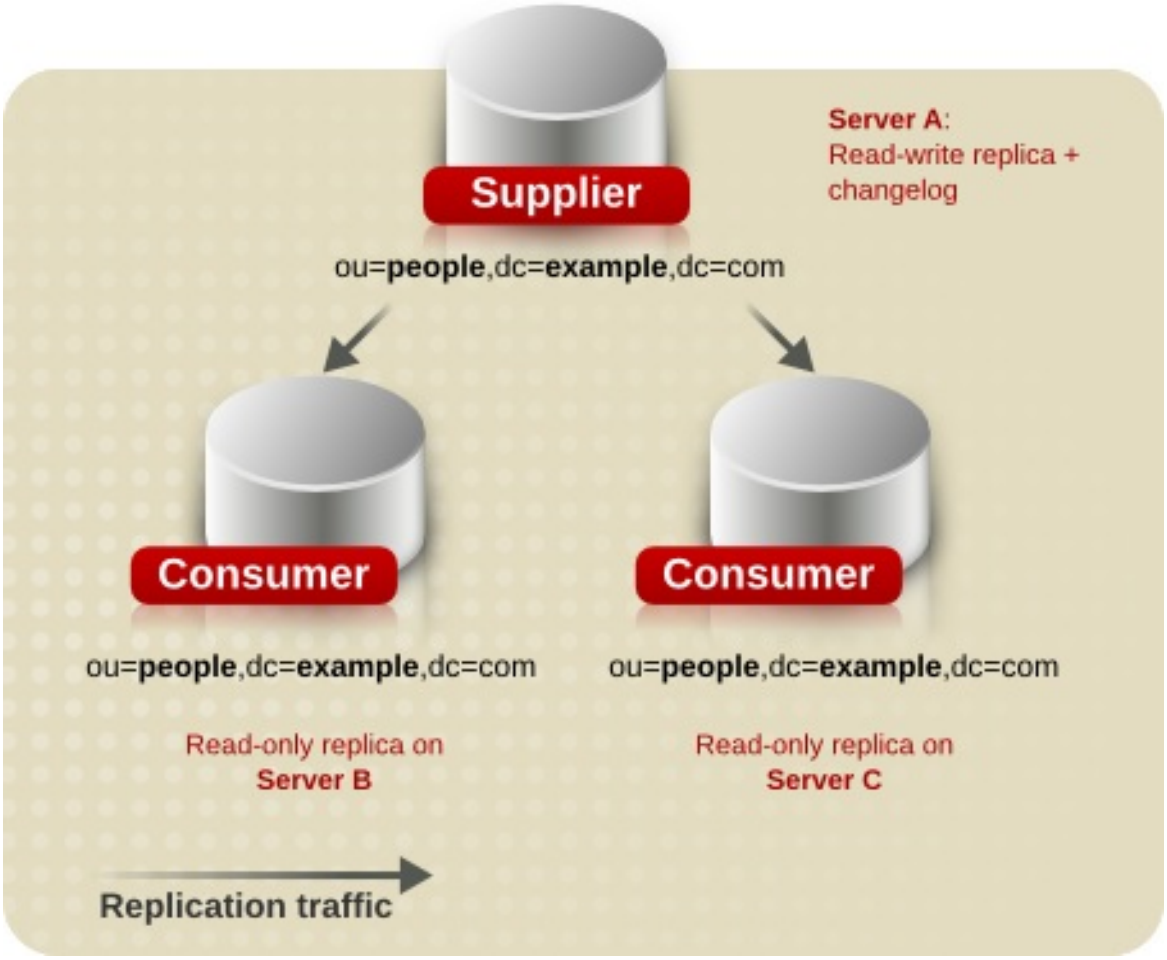


Figure 15.1. Single-Master Replication

In this particular configuration, the **ou=people,dc=example,dc=com** suffix receives a large number of search requests. Therefore, to distribute the load, this tree, which is mastered on Server A, is replicated to two read-only replicas located on Server B and Server C.

For information on setting up a single-master replication environment, see Section 15.5, "Configuring Single-Master Replication".

15.3.2. Multi-Master Replication

Directory Server also supports complex replication scenarios in which the same suffix (database) can be mastered on many servers. This suffix is held in a read-write replica on each server. This means that each server maintains a changelog for the read-write replica.

Multi-master replication in Directory Server supports as many as 20 masters, an unlimited number of hub suppliers, and an unlimited number of consumer servers. Each consumer server holds a read-only replica. The consumers can receive updates from any or all the suppliers. The consumers also have referrals defined for all the suppliers to forward any update requests that the consumers receive. Such scenarios are called *multi-master configurations*.

Figure 15.2, "Multi-Master Replication (Two Masters)" shows an example of multi-master replication scenario with two supplier servers and two consumer servers.

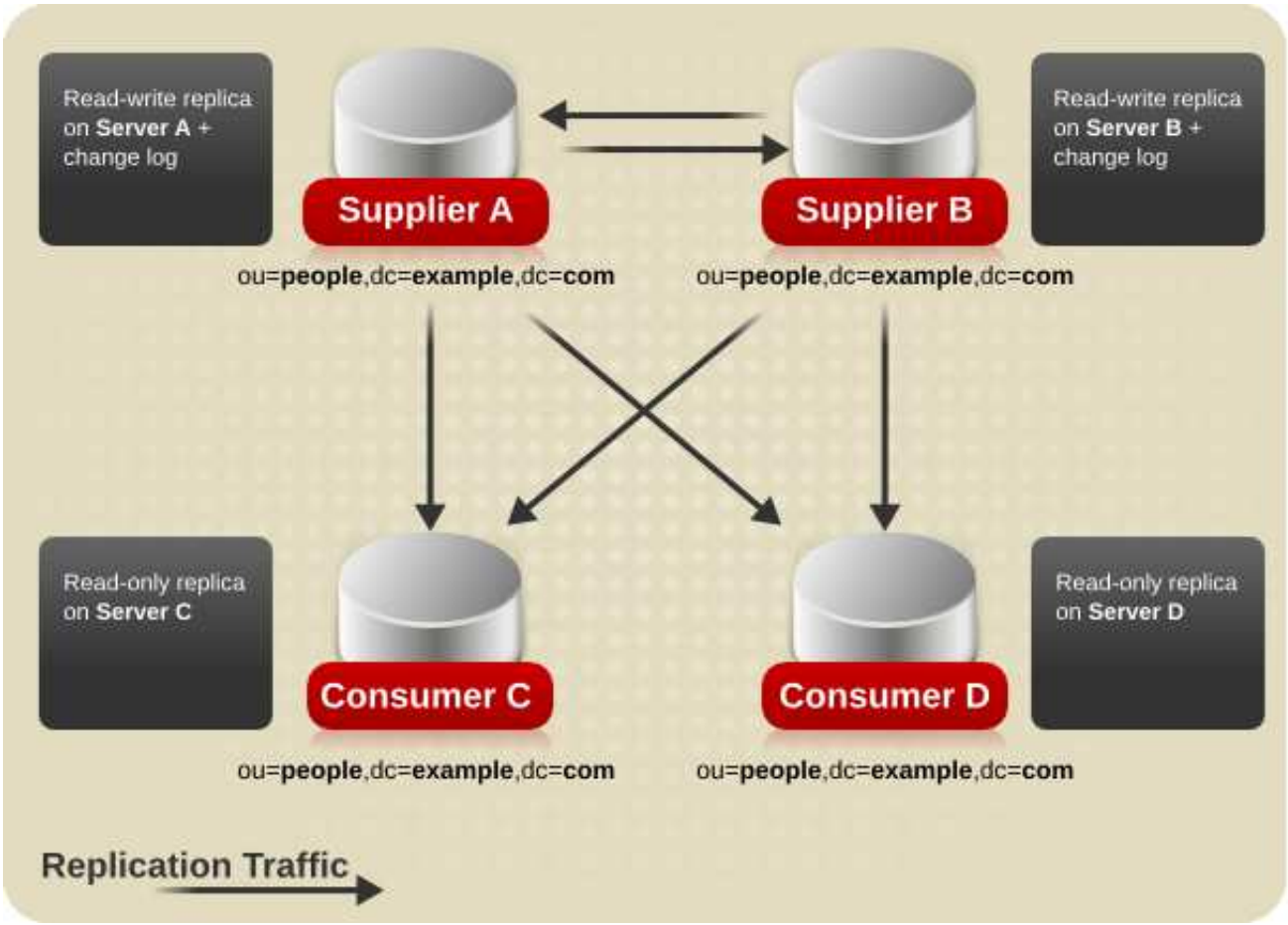


Figure 15.2. Multi-Master Replication (Two Masters)

Figure 15.3, "Multi-Master Replication (Four Masters)" shows a sample of multi-master replication scenario with four supplier servers and eight consumer servers. In this sample setup, each supplier server is configured with ten replication agreements to feed data to two other supplier servers and all eight consumer servers. (The Directory Server can have as many as 20 masters in a multi-master setup.)

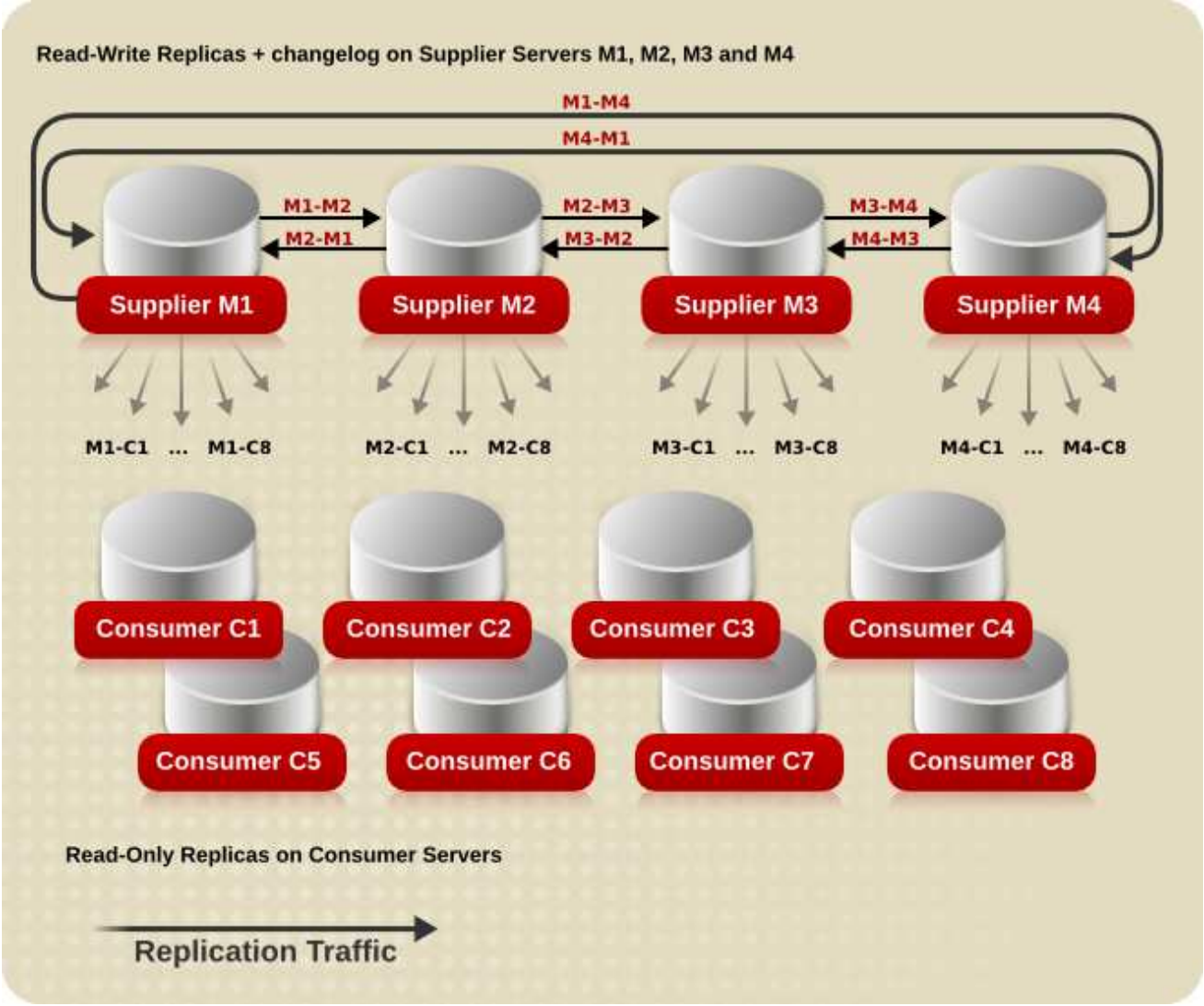
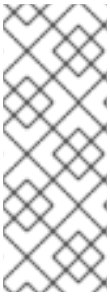


Figure 15.3. Multi-Master Replication (Four Masters)

Multi-master configurations have the following advantages:

- Automatic write failover when one supplier is inaccessible.
- Updates are made on a local supplier in a geographically distributed environment.



NOTE

The speed that replication proceeds depends on:

- The speed of the network.
- The number of outgoing and incoming replication agreements. Set up maximum 8 outbound and 4 inbound replication agreements for best performance.

For the procedure to set up multi-master replication, see Section 15.6, "Configuring Multi-Master Replication".

15.3.3. Cascading Replication

In a cascading replication scenario, one server, a *hub*, acts both as a consumer and a supplier. It holds a read-only replica and maintains a changelog, so it receives updates from the supplier server that holds

the master copy of the data and, in turn, supplies those updates to the consumer. Cascading replication is very useful for balancing heavy traffic loads or to keep master servers based locally in geographically-distributed environments.

Figure 15.4, "Cascading Replication" shows an example of a simple cascading replication scenario, though it is possible to create more complex scenarios with several hub servers.

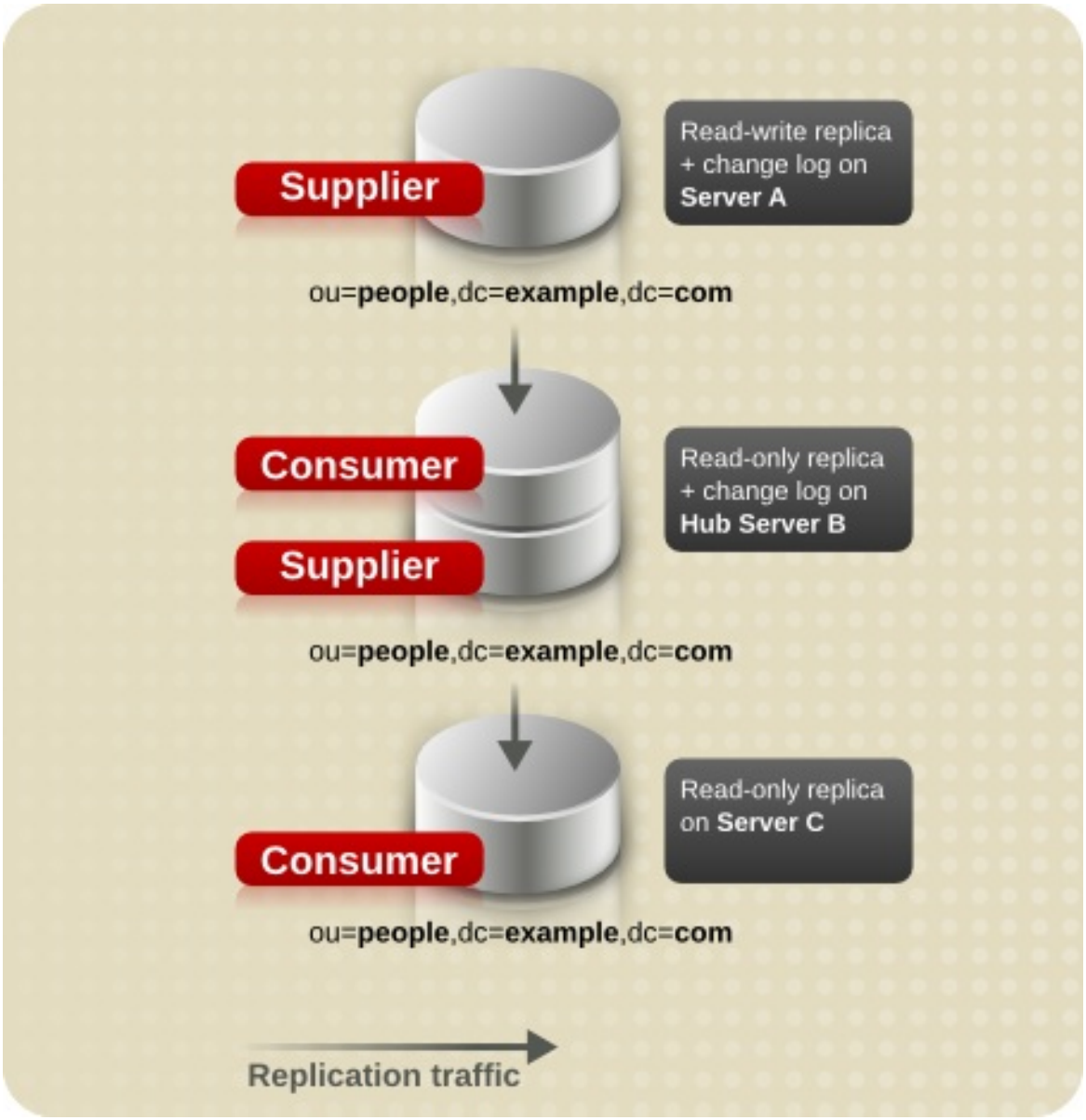


Figure 15.4. Cascading Replication

For information on setting up cascading replication, see Section 15.7, "Configuring Cascading Replication".

NOTE

Multi-master and cascading replication can be combined. For example, in the multi-master scenario illustrated in Figure 15.2, "Multi-Master Replication (Two Masters)", Server C and Server D could be hub servers that would replicate to any number of consumer servers.

15.4. CREATING THE SUPPLIER BIND DN ENTRY

A critical part of setting up replication is to create the entry, called the replication manager or supplier bind DN entry, that the suppliers use to bind to the consumer servers to perform replication updates.

The supplier bind DN must meet the following criteria:

- It must be unique.
- It must be created on the consumer server (or hub) and *not* on the supplier server.
- It must correspond to an actual entry on the consumer server.
- It must be created on *every* server that receives updates from another server.
- It must not be part of the replicated database for security reasons.
- It must be defined in the replication agreement on the supplier server.
- It must have an idle timeout period set to a high enough limit to allow the initialization process for large databases to complete. Using the ***nsIdleTimeOut*** operational attribute allows the replication manager entry to override the global ***nsslapd-idletimeout*** setting.

For example, the entry **cn=Replication Manager,cn=config** can be created under the **cn=config** tree on the consumer server. This would be the supplier bind DN that all supplier servers would use to bind to the consumer to perform replication operations.

NOTE

Avoid creating simple entries under the **cn=config** entry in the **dse.ldif** file. The **cn=cn=config** entry in the simple, flat **dse.ldif** configuration file is not stored in the same highly scalable database as regular entries. As a result, if many entries, and particularly entries that are likely to be updated frequently, are stored under **cn=config**, performance will suffer. However, although Red Hat recommends not storing simple user entries under **cn=config** for performance reasons, it can be useful to store special user entries such as the Directory Manager entry or replication manager (supplier bind DN) entry under **cn=config** since this centralizes configuration information.

On each server that acts as a consumer in replication agreements, create a special entry that the supplier will use to bind to the consumers. Make sure to create the entry with the attributes required by the authentication method specified in the replication agreement.

1. Stop the Directory Server. If the server is not stopped, the changes to the **dse.ldif** file will not be saved. See Section 1.4, “Starting and Stopping a Directory Server Instance” for more information on stopping the server.
2. Create a new entry, such as **cn=replication manager,cn=config**, in the **dse.ldif** file.
3. Specify a ***userPassword*** attribute-value pair.
4. Set an ***nsIdleTimeout*** period that gives the replication user a long enough time limit to allow replication initialization on large databases to complete.
5. If password expiration policy is enabled or ever will be enabled, disable it on the replication manager entry to prevent replication from failing due to passwords expiring. To disable the password expiration policy on the ***userPassword*** attribute, add the ***passwordExpirationTime*** attribute with a value of **20380119031407Z**, which means that the password will never expire.

6. Restart the Directory Server. See Section 1.4, "Starting and Stopping a Directory Server Instance" for more information on starting the server.

The final entry should resemble Example 15.4, "Example Supplier Bind DN Entry" .

Example 15.4. Example Supplier Bind DN Entry

```
dn: cn=replication manager,cn=config
objectClass: top
objectClass: device
objectClass: simpleSecurityObject
cn: replication manager
userPassword: strong_password
nsIdleTimeout: 0
```

When configuring a replica as a consumer, use the DN of this entry to define the supplier bind DN.

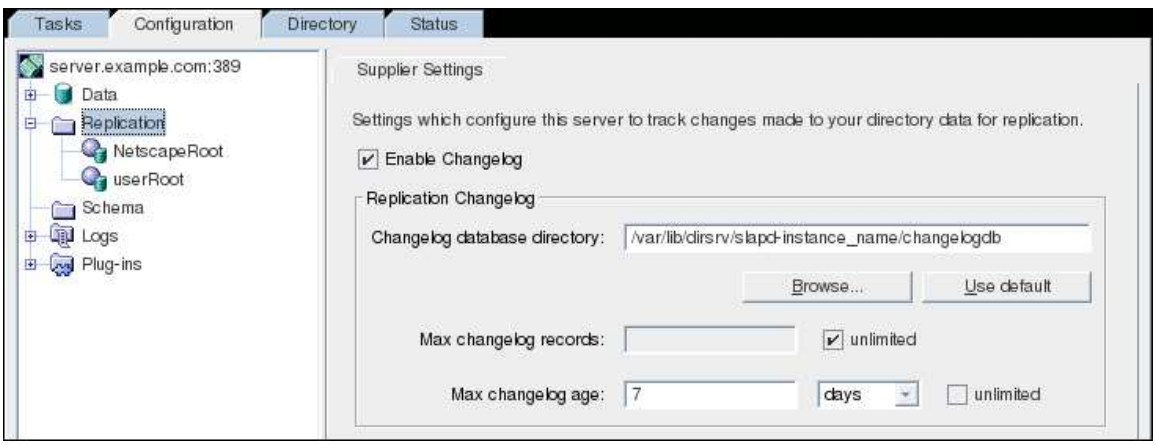
15.5. CONFIGURING SINGLE-MASTER REPLICATION

To set up single-master replication such as the configuration shown in Figure 15.1, "Single-Master Replication", between supplier Server A, which holds a read-write replica, and the two consumers Server B and Server C, which each hold a read-only replica, there are three major steps:

- Section 15.5.1, "Configuring the Read-Write Replica on the Supplier Server"
- Section 15.5.2, "Configuring the Read-Only Replica on the Consumer"
- Section 15.5.3, "Creating the Replication Agreement"

15.5.1. Configuring the Read-Write Replica on the Supplier Server

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



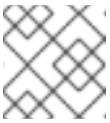
    4. Check the **Enable Changelog** check box.

This activates all of the fields in the pane below that were previously grayed out.

5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

6. Set the changelog parameters for the number and age of the log files.

   Clear the unlimited check boxes to specify different values.

   NOTE

   Red Hat recommends setting the maximum changelog age to **7 days**.

7. Click **Save**.

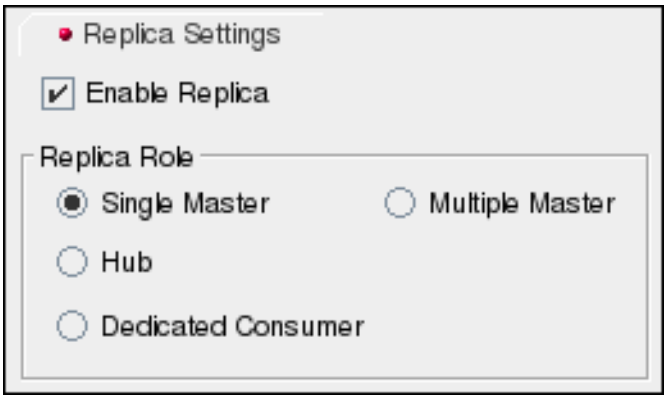
2. Specify the replication settings required for a read-write replica.

   1. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

      The **Replica Settings** tab opens in the right-hand side of the window.

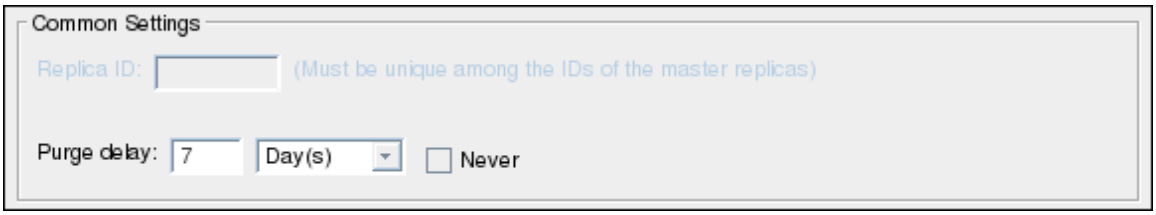
   2. Check the **Enable Replica** check box.

   3. In the **Replica Role** section, select the **Single Master** radio button.

   4. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.

      The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

   5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

      The purge delay is how often the state information stored for the replicated entries is deleted.

6. Click **Save**.

15.5.2. Configuring the Read-Only Replica on the Consumer

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 15.4, "Creating the Supplier Bind DN Entry" .

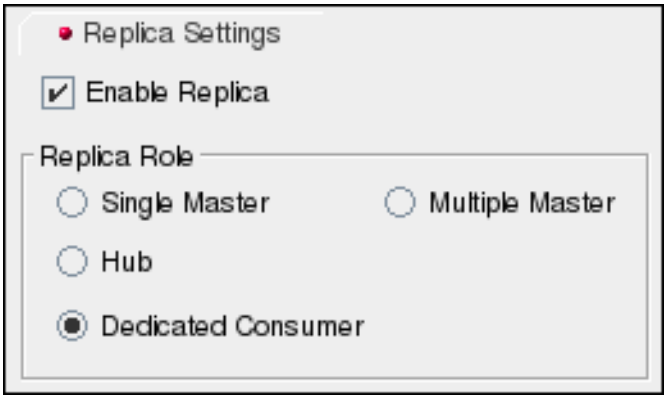
3. Specify the replication settings required for a read-only replica.

    1. In the Directory Server Console, select the **Configuration** tab.

    2. In the navigation tree, expand the **Replication** folder, and select the replica database.

        If you want to replicate the **o=NetscapeRoot** database, see Section 15.22, "Replicating o=NetscapeRoot for Administration Server Failover".

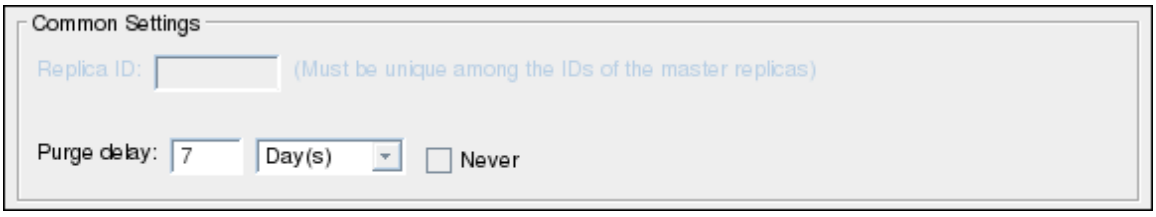
    3. In the **Replica Settings** tab of the selected database, check the **Enable Replica** check box.



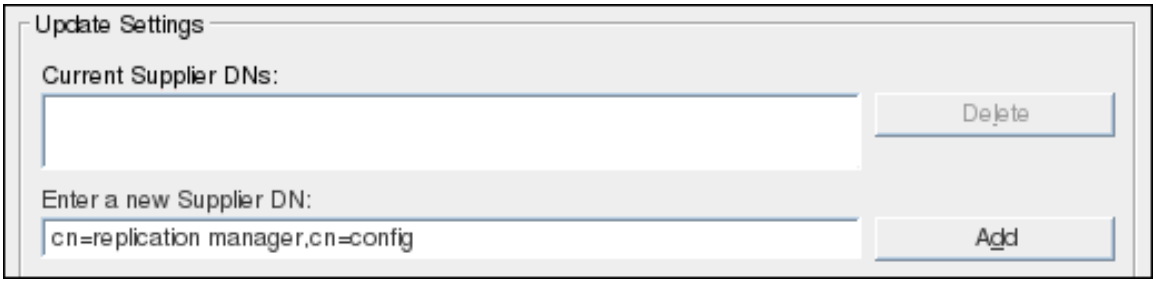
    4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.

    5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

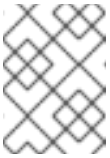
        This option indicates how often the state information stored for the replicated entries is purged.



    6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control.



NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using TLS, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.

NOTE

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

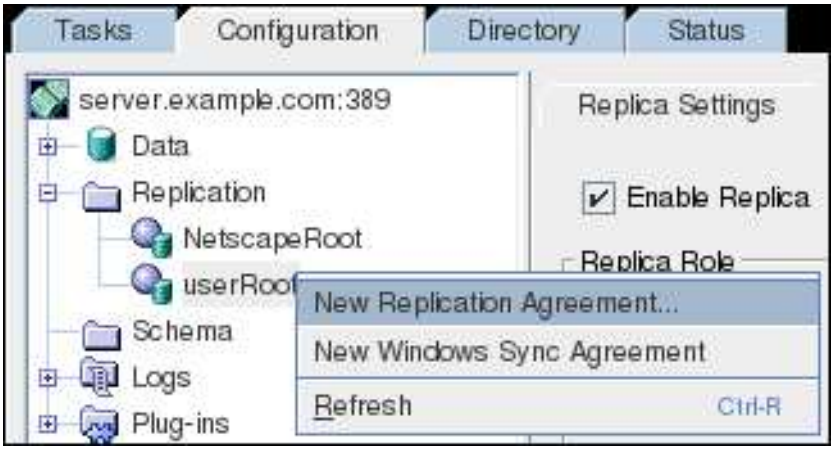
4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration.

15.5.3. Creating the Replication Agreement

On the supplier, create one replication agreement for each read-only replica. For example, in the scenario illustrated in Figure 15.1, “Single-Master Replication”, Server A has two replication agreements, one for Server B and one for Server C.

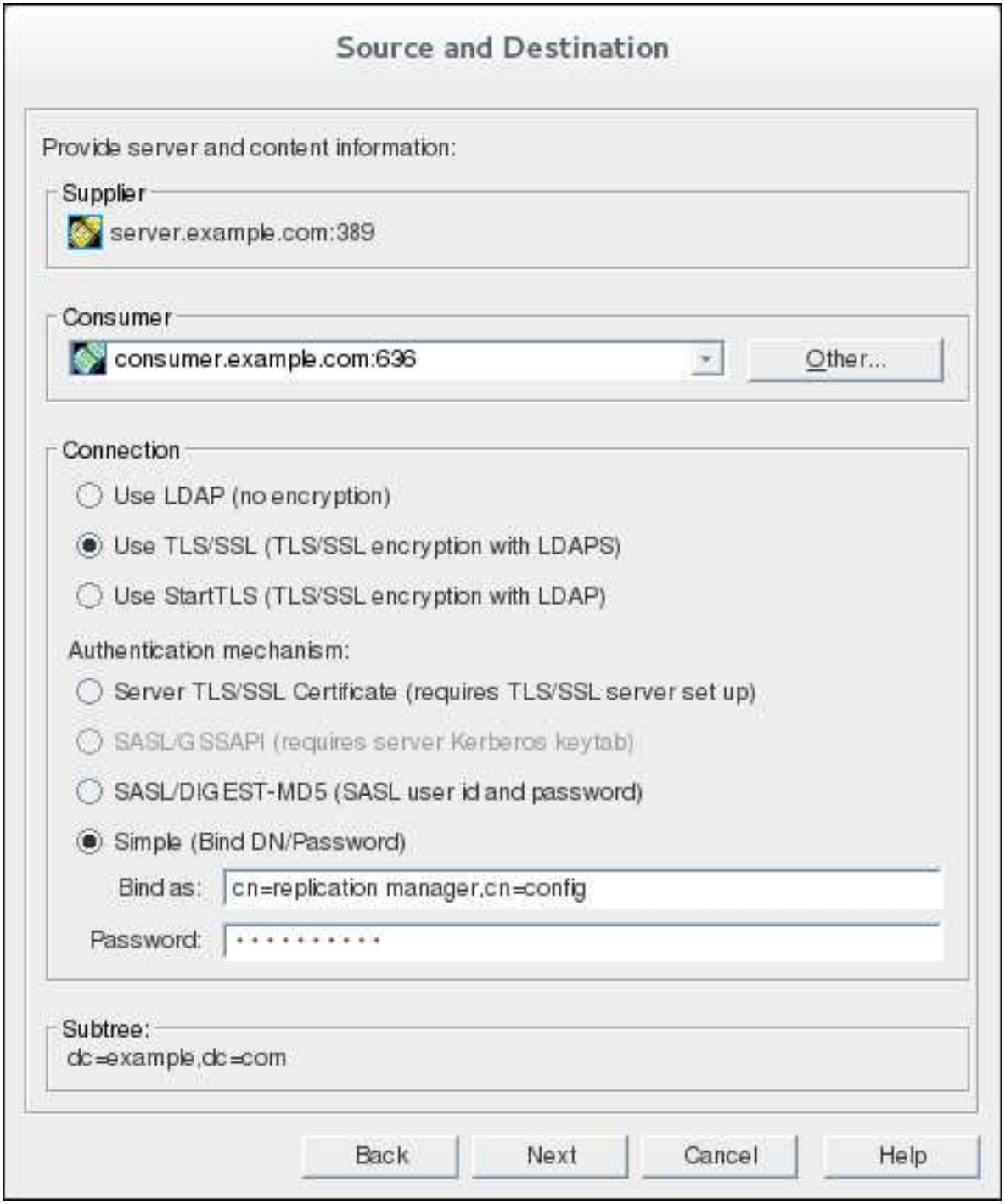
1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.



Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.

3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.



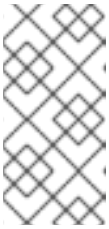
- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu.
- The port listed is the non-TLS port, even if the Directory Server instance is configured to run over TLS. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.
- If TLS is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for TLS client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

4. Select the connection type. There are three options:

   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.



NOTE

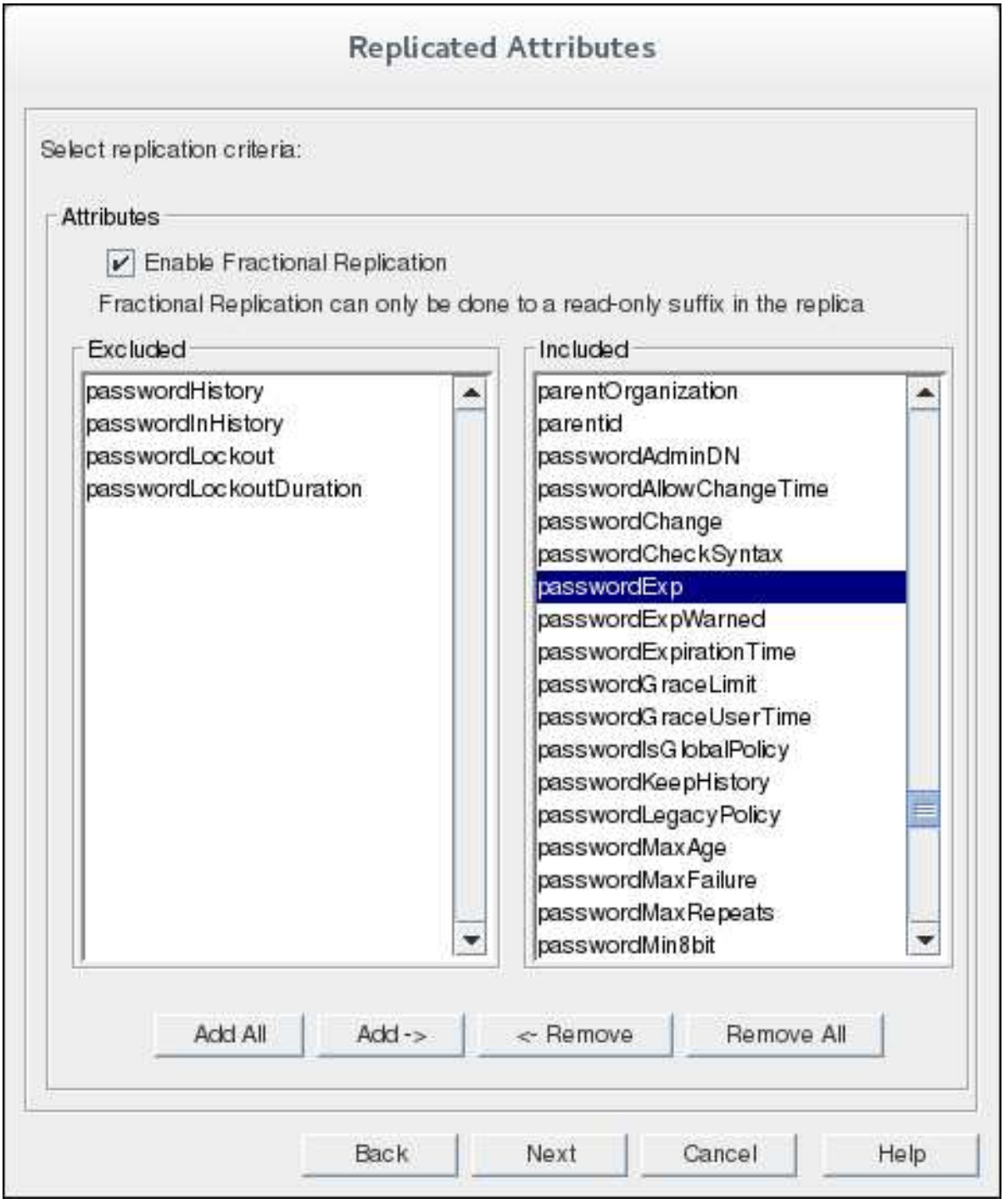
If secure binds are required for simple password authentication (Section 19.11.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (TLS and Start TLS connections or SASL authentication) is recommended, anyway.

5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.

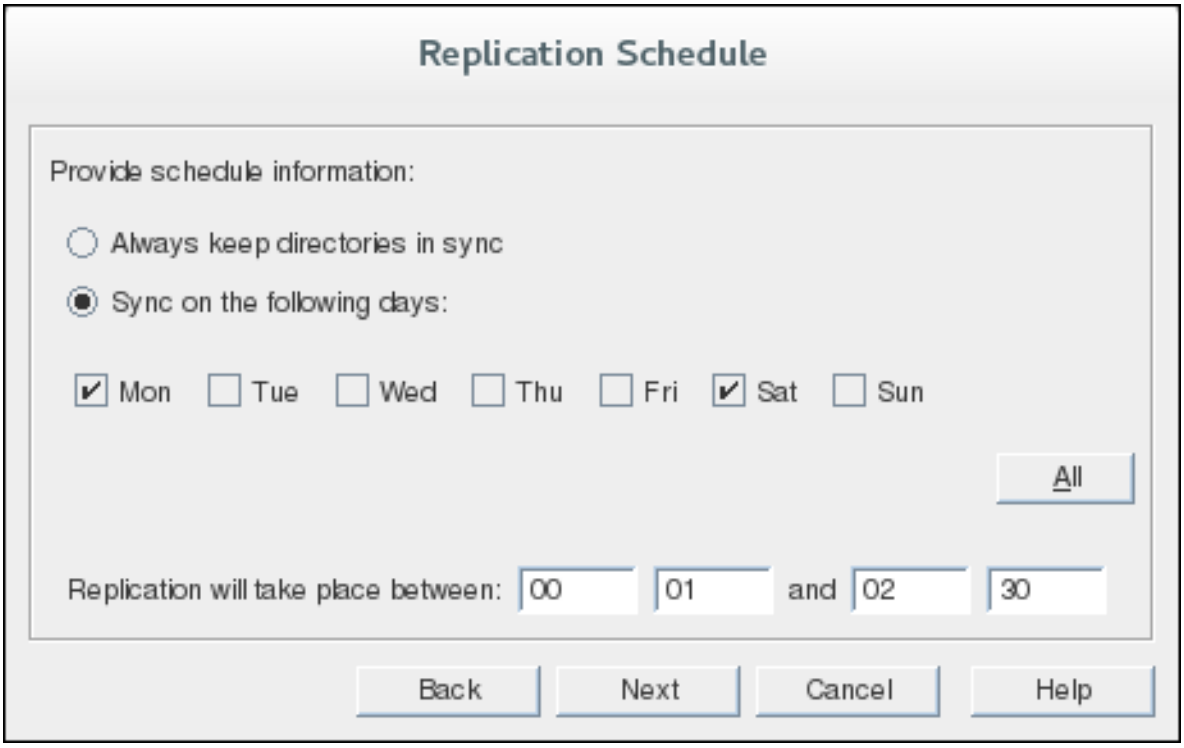
   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).
   - *Server TLS/SSL Certificate* uses the supplier's TLS certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

     Configuring TLS and certificate mapping is described in Section 9.4, "Enabling TLS".
   - *SASL/DIGEST-MD5*, like simple authentication, this insecure method requires only the bind DN and password to authenticate. This can run over a standard or TLS connection.
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 9.10.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 9.9.3.1, "Configuring SASL Identity Mapping from the Console").

6. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



7. Set the schedule for when replication runs. By default, replication runs continually.

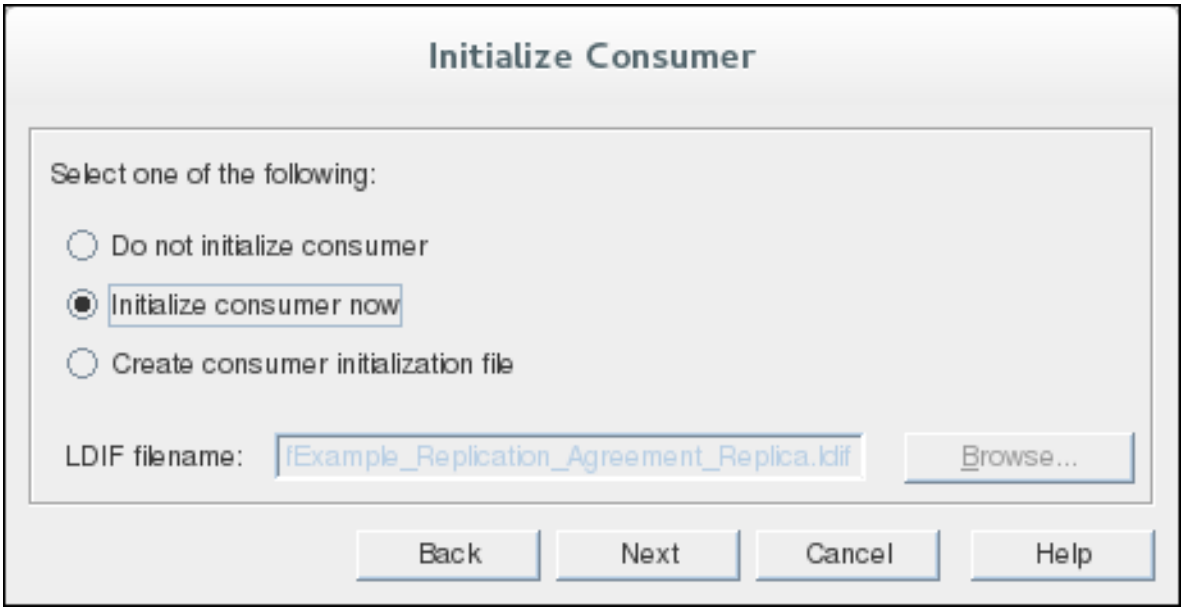


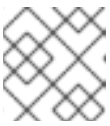
NOTE

The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

8. Select **Initialize consumer now**, to start initializing after the replication agreement was completed, and click **Next**.



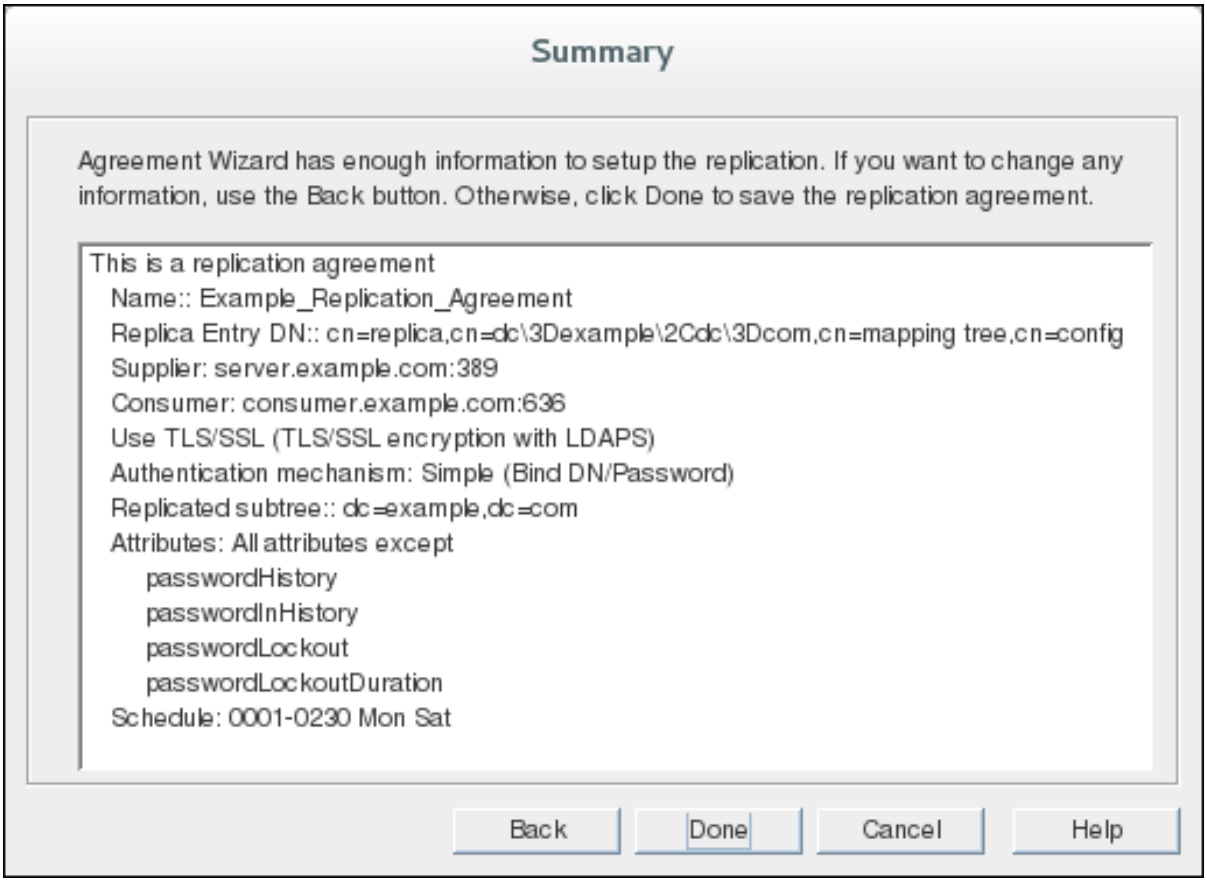


NOTE

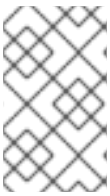
Replication *will not* begin until the consumer is initialized.

For further details on initializing consumers, see Section 15.18, “Initializing Consumers” .

9. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.



The replication agreement is set up.



NOTE

After creating a replication agreement, the connection type (TLS or non-TLS) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

15.6. CONFIGURING MULTI-MASTER REPLICATION

In a multi-master configuration, many suppliers can accept updates, synchronize with each other, and update all consumers. The consumers can send referrals for updates to all masters.

Directory Server supports 20-way multi-master replication, meaning that there can be up to 20 masters (and an unlimited number of hub suppliers) in a single replication scenario. Directory Server allows an unlimited number of consumers.

To set up multi-master replication, set up all of the consumers first, then set up the suppliers, and last, initialize all of the databases.

- Section 15.6.1, “Configuring the Read-Write Replicas on the Supplier Servers”
- Section 15.6.2, “Configuring the Read-Only Replicas on the Consumer Servers”
- Section 15.6.3, “Setting up the Replication Agreements”

- Section 15.6.4, "Preventing Monopolization of a Consumer in Multi-Master Replication"

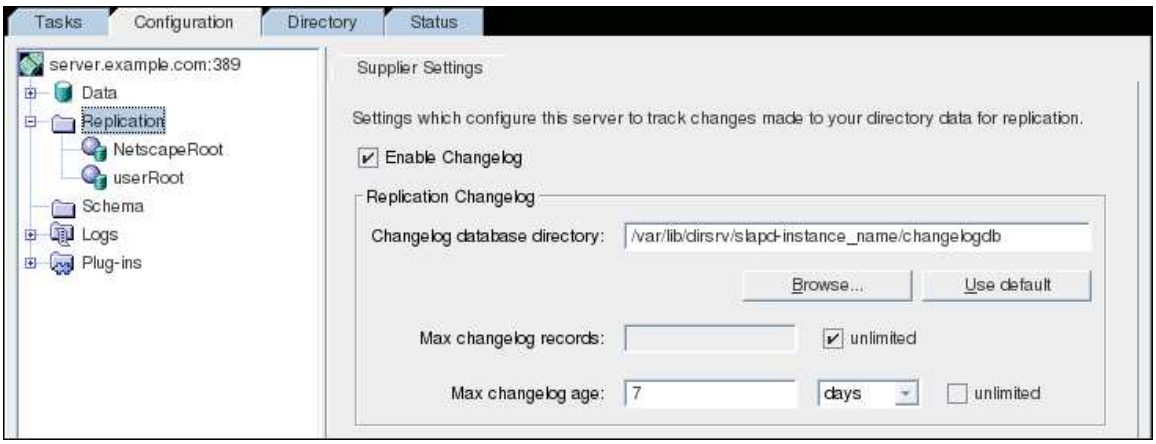
NOTE

More than 10 databases running with replication or more than on a supplier can cause performance degradation. To support that many consumers, introduce hub replicas between the suppliers and consumers. See Section 15.7, "Configuring Cascading Replication".

15.6.1. Configuring the Read-Write Replicas on the Supplier Servers

Set up each supplier server. The first supplier configured should be used to initialize the other suppliers in the multi-master replication environment.

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



    4. Check the **Enable Changelog** check box.

       This activates all of the fields in the pane below that were previously grayed out.

    5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

    6. Set the changelog parameters for the number and age of the log files.

       Clear the unlimited check boxes to specify different values.

    7. Click **Save**.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. This is the special entry that the other suppliers will use to bind to this supplier, as in other supplier-consumer relationships. This is described in Section 15.4, "Creating the Supplier Bind DN Entry" .

NOTE

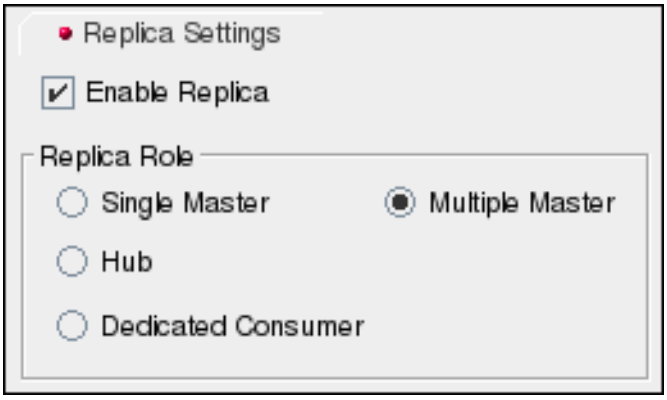
For multi-master replication, it is necessary to create this supplier bind DN on the supplier servers as well as the consumers because the suppliers act as both consumer and supplier to the other supplier servers.

3. Specify the replication settings for the multi-mastered read-write replica.

    1. In the Directory Server Console, select the **Configuration** tab.

    2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

        The **Replica Settings** tab for that database opens in the right-hand side of the window.



    3. Check the **Enable Replica** check box.

    4. In the **Replica Role** section, select the **Multiple Master** radio button.

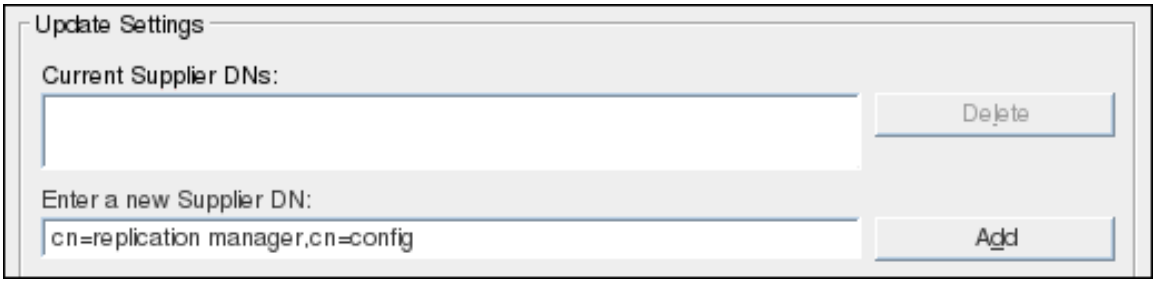
    5. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.

        The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

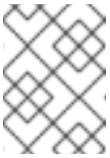
    6. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

        The purge delay is how often the state information stored for the replicated entries is deleted.

    7. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.



NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

8. Specify the LDAP URL (*ldap://hostname:port* or *ldap://IP_address:port*, with IPv4 or IPv6 addresses) for any supplier servers to which to refer updates, such as the other suppliers in the multi-master replication set. Only specify the URL for the supplier server.

   For clients to bind using TLS, specify a URL beginning with **ldaps:**//.

9. Click **Save**.

15.6.2. Configuring the Read-Only Replicas on the Consumer Servers

First, configure *every* consumer before creating any replication agreements.

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

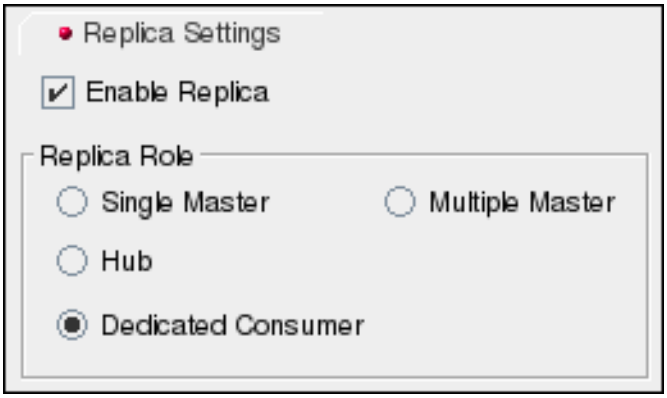
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 15.4, "Creating the Supplier Bind DN Entry" .

3. Specify the replication settings required for a read-only replica.

   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

      The **Replica Settings** tab for that database opens in the right-hand side of the window.

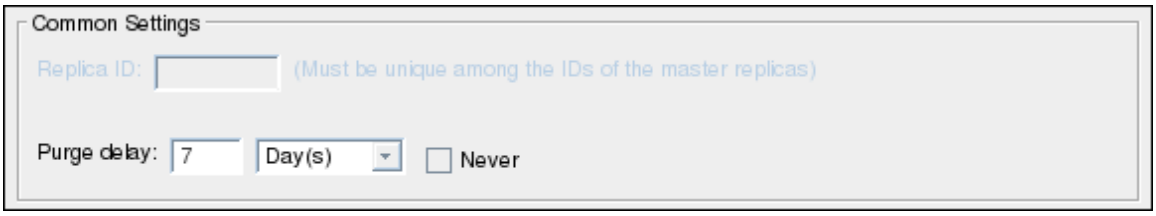


3. Check the **Enable Replica** check box.

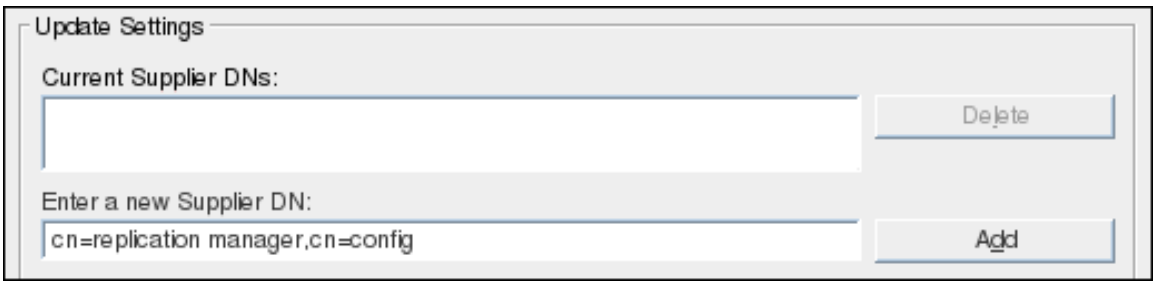
4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   This option indicates how often the state information stored for the replicated entries is purged.



6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



   The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

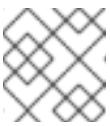
NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using TLS, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.



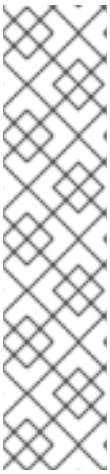
NOTE

It is also possible to use IPv4 or IPv6 addresses instead of the host name.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration.

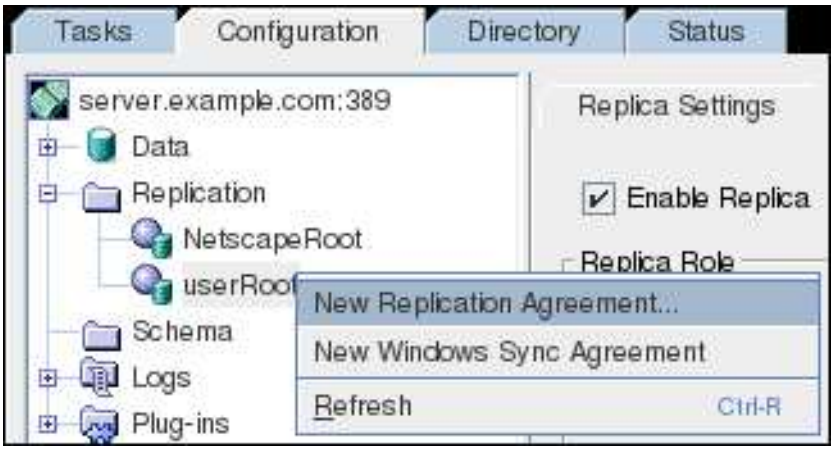
15.6.3. Setting up the Replication Agreements



NOTE

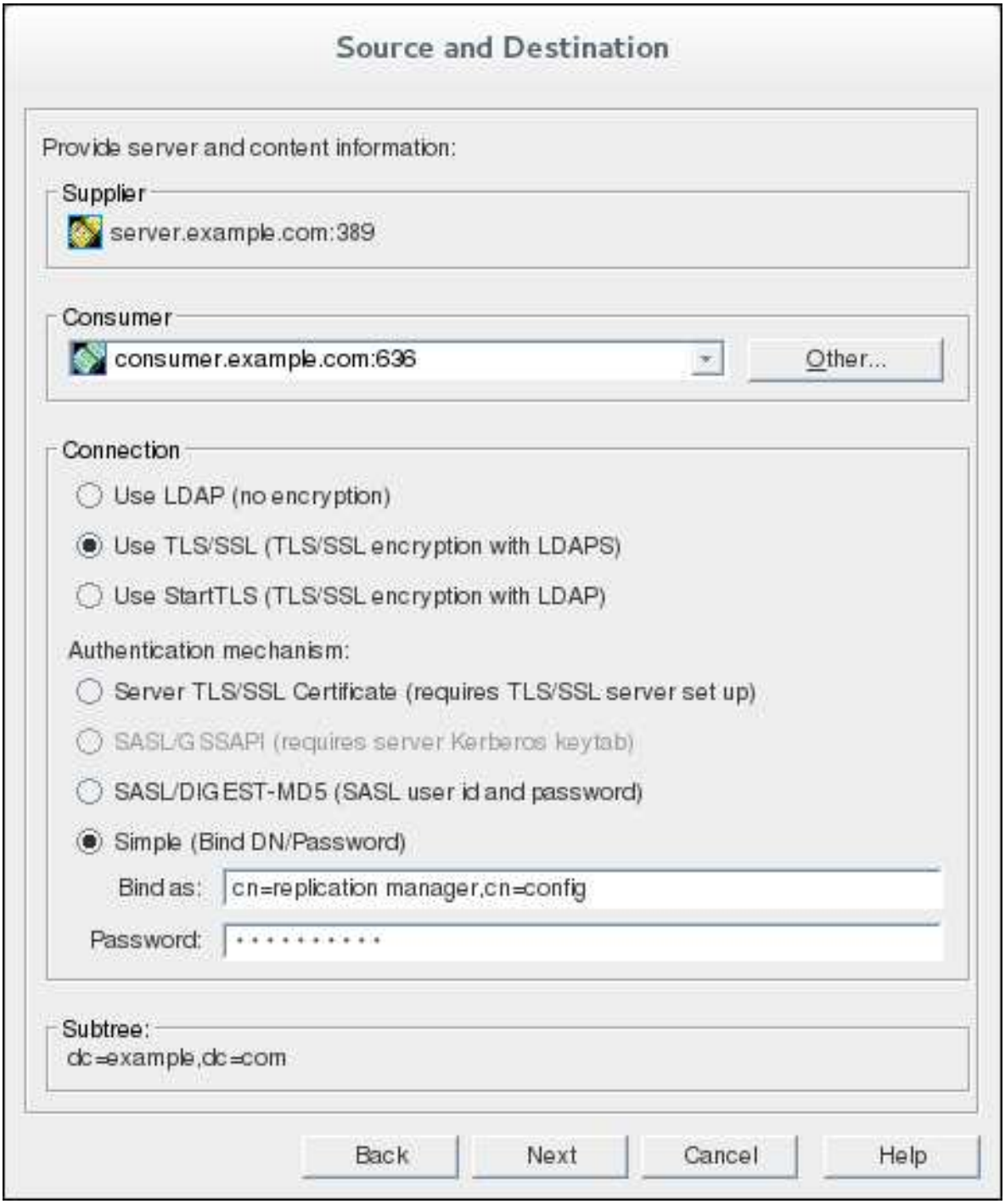
1. First set up replication agreements on a single supplier, the data master, between the other multi-master suppliers, and initialize all of the other suppliers.
2. Then create replication agreements for all other suppliers in the multi-master replication set, but *do not* reinitialize any of the suppliers.
3. Then create replication agreements for all of the consumers from the single data master, and initialize the consumers.
4. Then create replication agreements for all of the consumers from for all of the other suppliers, but *do not* reinitialize any of the consumers.

1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.



   Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.
3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.



- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu. The server URL can be entered manually, in the format *hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses.

- The port listed is the non-TLS port, even if the Directory Server instance is configured to run over TLS. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.

- If TLS is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for TLS client authentication. Otherwise, fill in the supplier bind DN and password.

NOTE

If attribute encryption is enabled, a secure connection is required for the encrypted attributes to be replicated.

4. Select the connection type. There are three options:

   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

If secure binds are required for simple password authentication (Section 19.11.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (TLS and Start TLS connections or SASL authentication) is recommended, anyway.

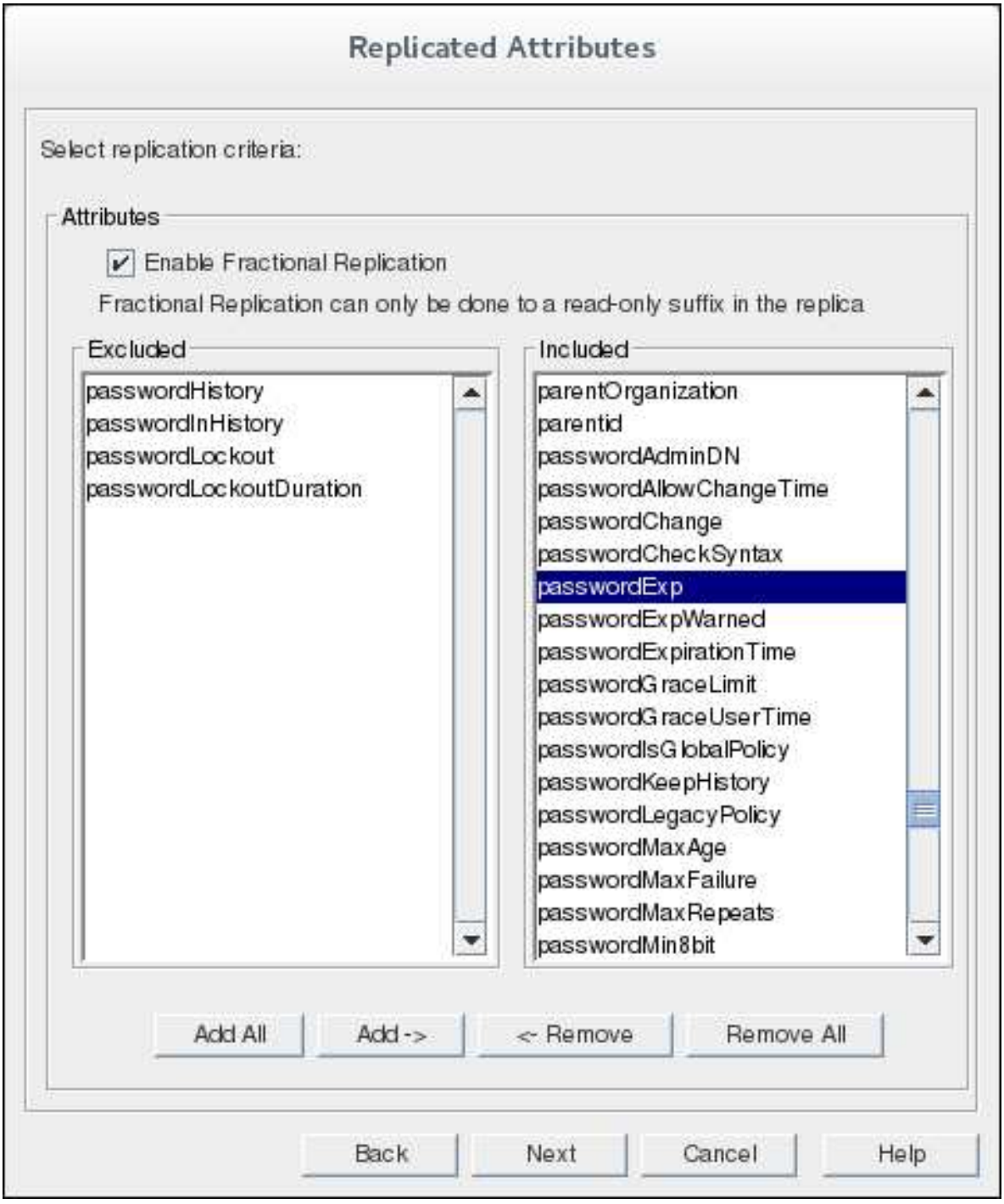
5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.

   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).
   - *Server TLS/SSL Certificate* uses the supplier's TLS certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

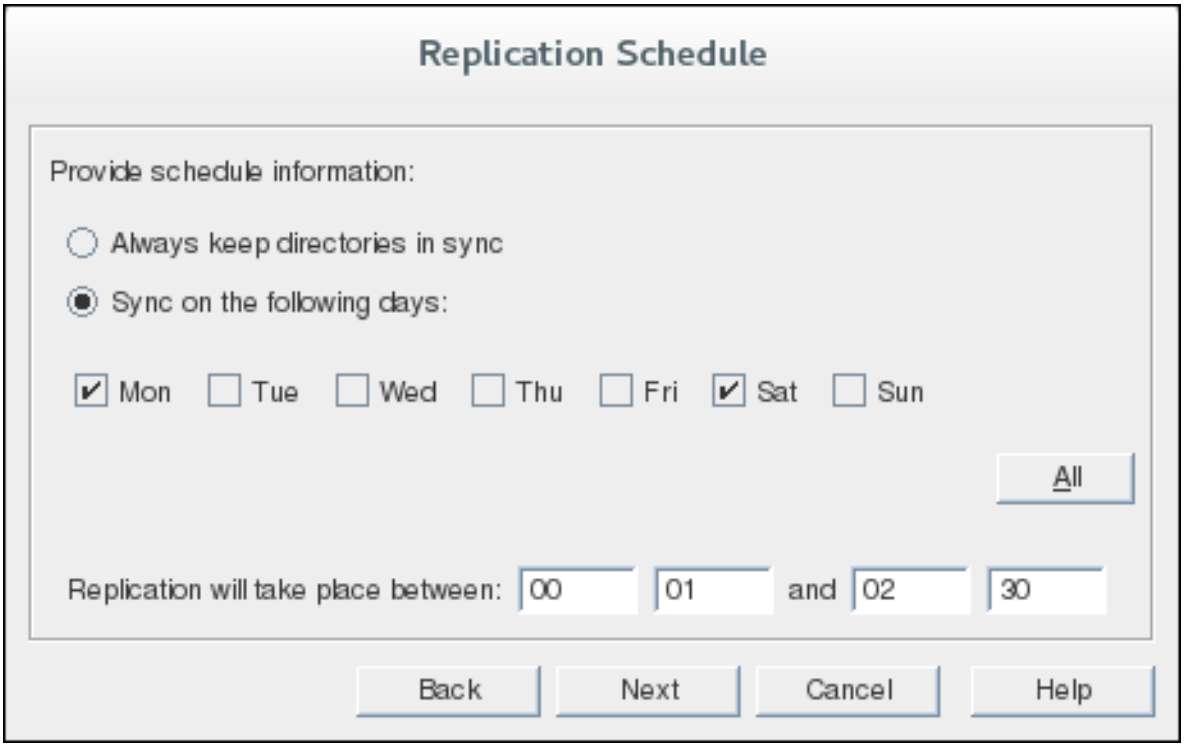
     Configuring TLS and certificate mapping is described in Section 9.4, "Enabling TLS".
   - *SASL/DIGEST-MD5*, like simple authentication, requires only the bind DN and password to authenticate. This can run over a standard or TLS connection.
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 9.10.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 9.9.3.1, "Configuring SASL Identity Mapping from the Console").

6. Hit **Next**.

7. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



8. Set the schedule for when replication runs. By default, replication runs continually.



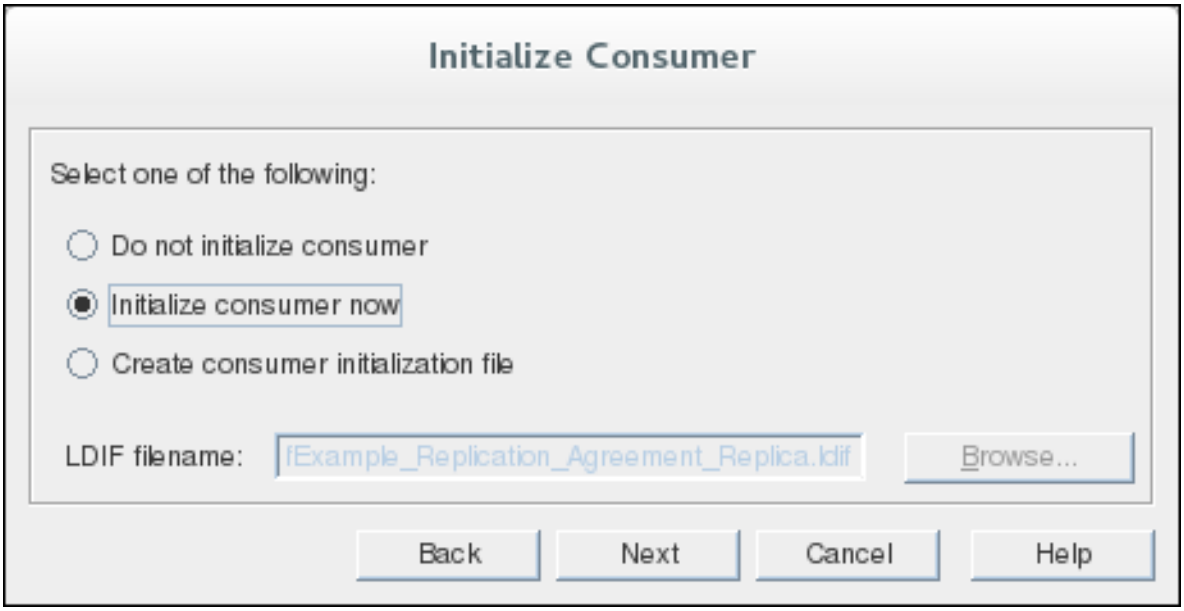
NOTE

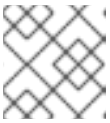
The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

9. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see Section 15.18, "Initializing Consumers". For multi-master replication, consider the following:

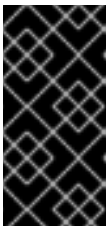
   - Ensure one supplier has the complete set of data to replicate to the other suppliers. Use this one supplier to initialize the replica on all other suppliers in the multi-master replication set.
   - Initialize the replicas on the consumer servers from any of the multi-master suppliers.
   - Do not try to *reinitialize* the servers when the replication agreements are set up. For example, do not initialize server1 from server2 if server2 has already been initialized from server1. In this case, select **Do not initialize consumer**.





NOTE

Replication *will not* begin until the consumer is initialized.

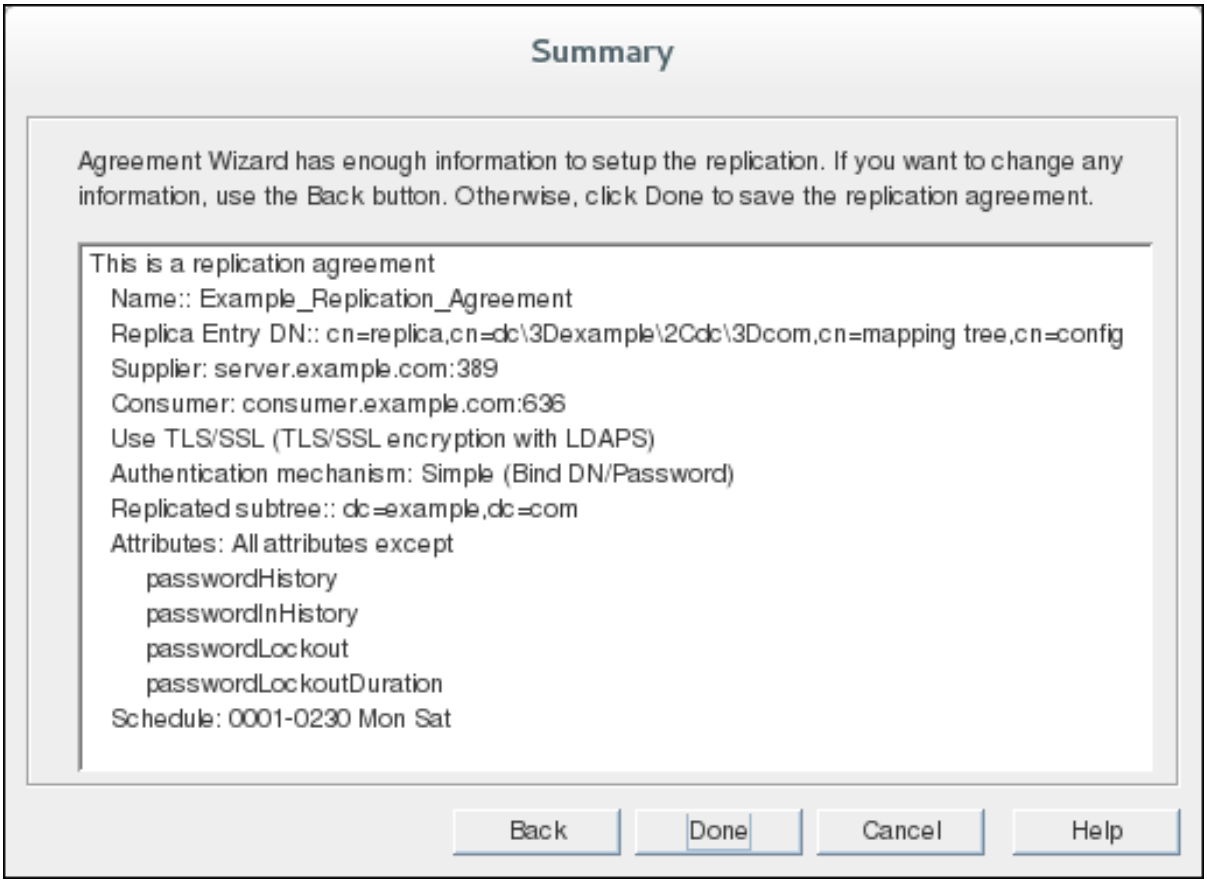


IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Hit **Next**.

10. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.



The replication agreement is set up.

NOTE

At the end of this procedure, all supplier servers will have mutual replication agreements, which means that they can accept updates from each other.

NOTE

After creating a replication agreement, the connection type (TLS or non-TLS) cannot be changed because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

15.6.4. Preventing Monopolization of a Consumer in Multi-Master Replication

One of the features of multi-master replication is that a supplier acquires exclusive access to the consumer for the replicated area. During this time, other suppliers are locked out of direct contact with the consumer. If a supplier attempts to acquire access while locked out, the consumer sends back a busy response, and the supplier sleeps for several seconds before making another attempt. During a low update load, the supplier sends its update to another consumer while the first consumer is locked and then send updates when the first consumer is free again.

A problem can arise if the locking supplier is under a heavy update load or has a lot of pending updates in the changelog. If the locking supplier finishes sending updates and then has more pending changes to send, it will immediately attempt to reacquire the consumer and will most likely succeed, since the other suppliers usually will be sleeping. This can cause a single supplier to monopolize a consumer for several hours or longer.

The following attributes address this issue:

nsds5ReplicaBusyWaitTime

Sets the time in seconds a supplier waits after a consumer sends back a busy response before making another attempt to acquire access.

For example, to configure that a supplier waits **5** seconds before making another acquire attempt:

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

dn: cn=Replication_Agreement_Name,cn=replica,cn=suffix_Name,cn=mapping tree,cn=config
changetype: modify
replace: nsds5ReplicaBusyWaitTime
nsds5ReplicaBusyWaitTime: 5
```

nsds5ReplicaSessionPauseTime

Sets the time in seconds a supplier waits between two update sessions. If you set a value lower or equal than the value specified in ***nsds5ReplicaBusyWaitTime***, Directory Server automatically uses a value for the ***nsds5ReplicaSessionPauseTime*** parameter, that is one second higher than the value set in ***nsds5ReplicaBusyWaitTime***.

For example, to configure that the supplier waits **10** seconds between two update sessions:

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

dn: cn=Replication_Agreement_Name,cn=replica,cn=suffix_Name,cn=mapping tree,cn=config
changetype: modify
replace: nsds5ReplicaSessionPauseTime
nsds5ReplicaSessionPauseTime: 10
```

nsds5ReplicaReleaseTimeout

Sets the timeout after which a master will release the replica, whether or not it has finished sending its updates. This prevents a single master from monopolizing a replica.

For example, to configure in a heavy replication environment that a master will release a replica after **90** seconds:

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

dn: cn=replica,cn=suffix_Name,cn=mapping tree,cn=config
changetype: modify
replace: nsds5ReplicaReleaseTimeout
nsds5ReplicaReleaseTimeout: 90
```

For further details, see the parameter descriptions in the *Red Hat Directory Server Configuration, Command, and File Reference*.

To log replica busy errors, enable **Replication** error logging (log level **8192**). See Section 20.3.7, "Configuring Log Levels".

15.7. CONFIGURING CASCADING REPLICATION

Setting up cascading replication, as shown in Figure 15.4, "Cascading Replication", has three major steps, for each server in the scenario, the supplier on Server A, which holds a read-write replica; the

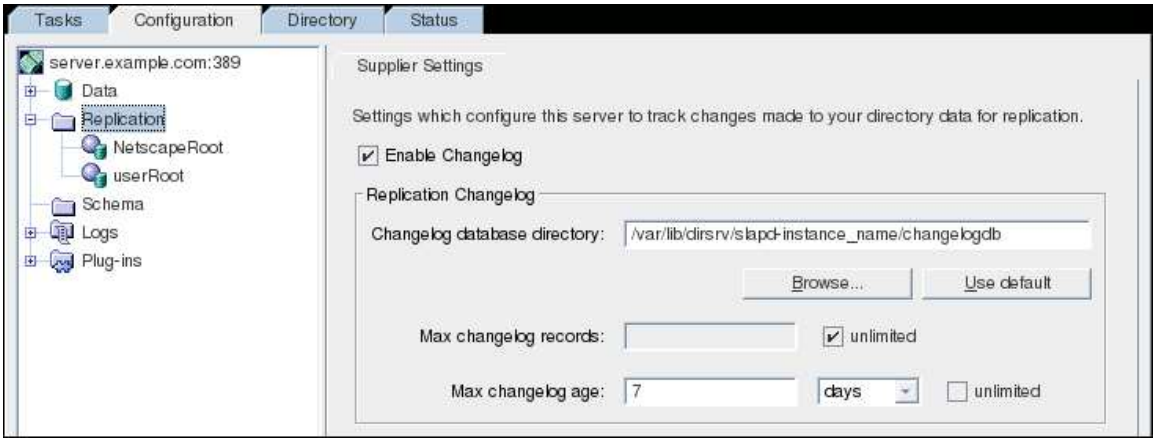
consumer/supplier on hub Server B, which holds a read-only replica; and the consumer on Server C, which holds a read-only replica:

- Section 15.7.1, "Configuring the Read-Write Replica on the Supplier Server"
- Section 15.7.2, "Configuring the Read-Only Replica on the Consumer Server"
- Section 15.7.3, "Configuring the Read-Only Replica on the Hub"
- Section 15.7.4, "Setting up the Replication Agreements"

15.7.1. Configuring the Read-Write Replica on the Supplier Server

Next, configure the supplier server, which holds the original copy of the database:

1. Specify the supplier settings for the server.
    1. In the Directory Server Console, select the **Configuration** tab.
    2. In the navigation tree, select the **Replication** folder.
    3. In the right-hand side of the window, select the **Supplier Settings** tab.



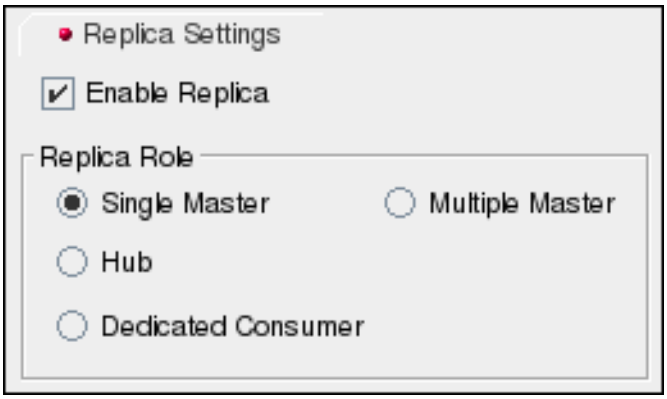
    4. Check the **Enable Changelog** check box.

       This activates all of the fields in the pane below that were previously grayed out.
    5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.
    6. Set the changelog parameters for the number and age of the log files.

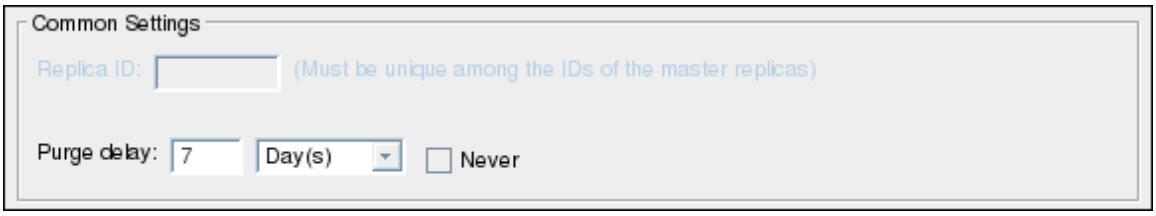
       Clear the unlimited check boxes to specify different values.
    7. Click **Save**.
2. Specify the replication settings required for a read-write replica.
    1. In the navigation tree on the **Configuration** tab, expand the **Replication** node, and highlight the database to replicate.

       The **Replica Settings** tab opens in the right-hand side of the window.
    2. Check the **Enable Replica** check box.

3. In the **Replica Role** section, select the **Single Master** radio button.



4. In the **Common Settings** section, specify a **Replica ID**, which is an integer between **1** and **65534**, inclusive.



   The replica ID must be unique for a given suffix, different from any other ID used for read-write replicas on this server and on other servers.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

   The purge delay is how often the state information stored for the replicated entries is deleted.

6. Click **Save**.

After setting up the supplier replica, begin configuring the replication agreements.

15.7.2. Configuring the Read-Only Replica on the Consumer Server

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

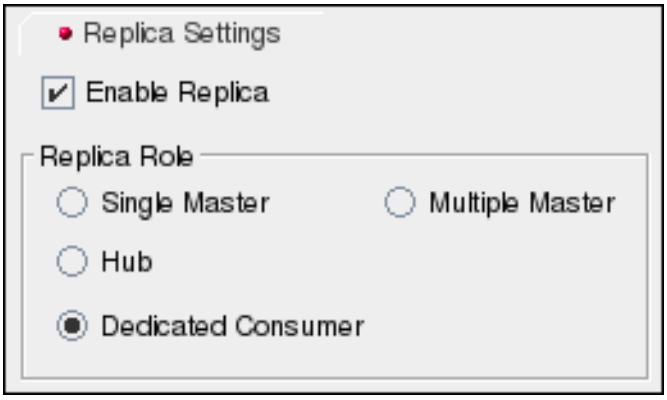
2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 15.4, "Creating the Supplier Bind DN Entry" .

3. Specify the replication settings required for a read-only replica.

   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

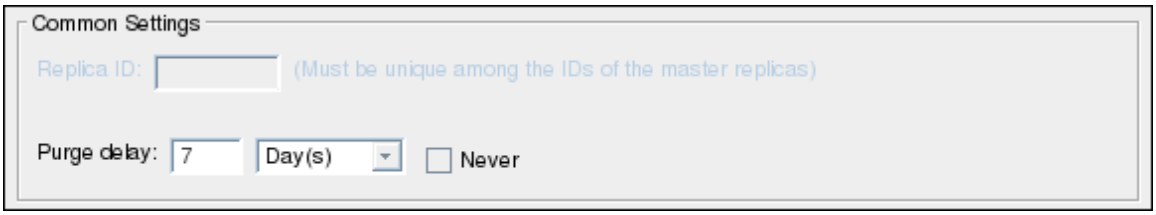
      The **Replica Settings** tab for that database opens in the right-hand side of the window.



3. Check the **Enable Replica** check box.

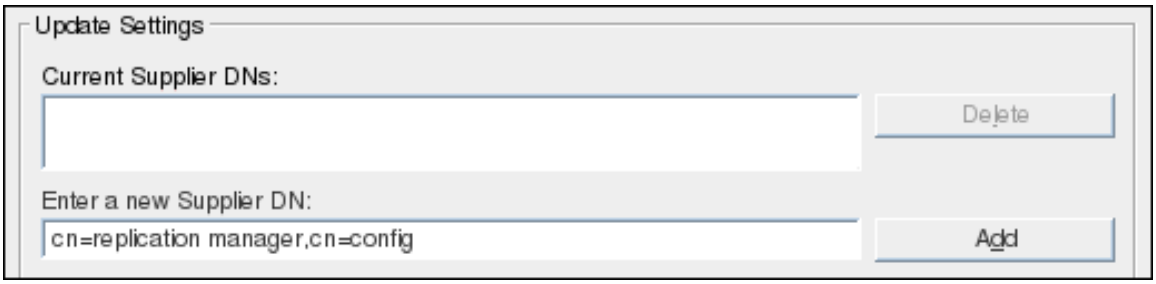
4. In the **Replica Role** section, select the **Dedicated Consumer** radio button.

5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.

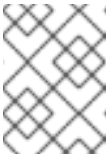


This option indicates how often the state information stored for the replicated entries is purged.

6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.



NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for any supplier servers to which to refer updates.

By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

In cascading replication, referrals are automatically sent to the hub server, which in turn refers the request to the original supplier. Therefore, set a referral to the original supplier to replace the automatically generated referral.

4. Click **Save**.

Repeat these steps for every consumer server in the replication configuration, then configure the hub replica.

15.7.3. Configuring the Read-Only Replica on the Hub

Do this to set up a hub, which receives replication updates from the supplier and propagates them to consumers:

1. Create the database for the read-only replica if it does not exist. See Section 2.1.1, "Creating Suffixes" for instructions on creating suffixes.

2. Create the entry for the supplier bind DN on the consumer server if it does not exist. The supplier bind DN is the special entry that the supplier will use to bind to the consumer. This is described in Section 15.4, "Creating the Supplier Bind DN Entry" .

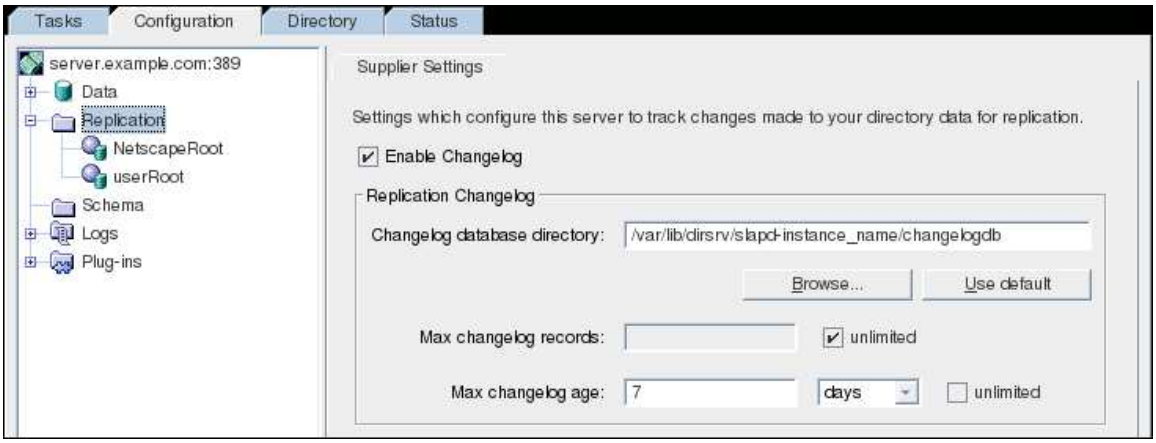
3. Create the changelog for the hub server.

   The hub must maintain a changelog even though it does not accept update operations because it records the changes sent from the supplier server.

   1. In the Directory Server Console, select the **Configuration** tab.

   2. In the navigation tree, select the **Replication** folder.

   3. In the right-hand side of the window, select the **Supplier Settings** tab.



   4. Check the **Enable Changelog** check box.

      This activates all of the fields in the pane below that were previously grayed out.

   5. Specify a changelog by clicking the **Use default** button, or click the **Browse** button to display a file selector.

   6. Set the changelog parameters for the number and age of the log files.

      Clear the unlimited check boxes to specify different values.

   7. Click **Save**.

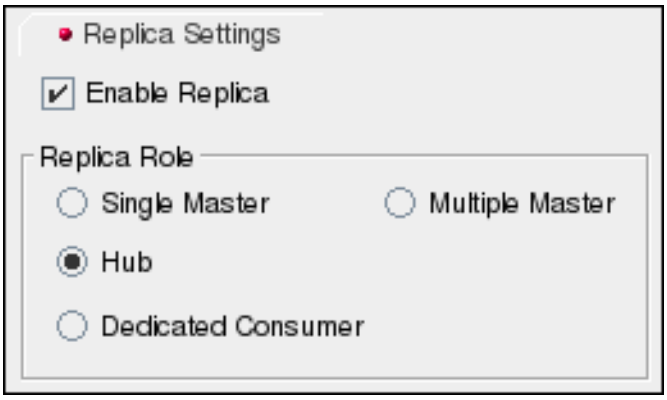
4. Specify the required hub replica settings.

    1. In the Directory Server Console, select the **Configuration** tab.

    2. In the navigation tree, expand the **Replication** folder, and highlight the replica database.

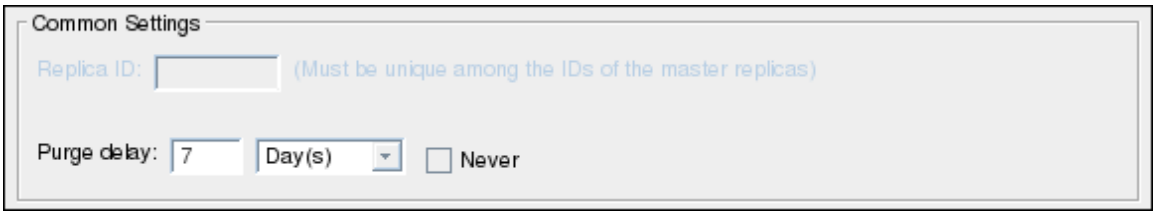
        The **Replica Settings** tab for that database opens in the right-hand side of the window.

    3. Check the **Enable Replica** check box.



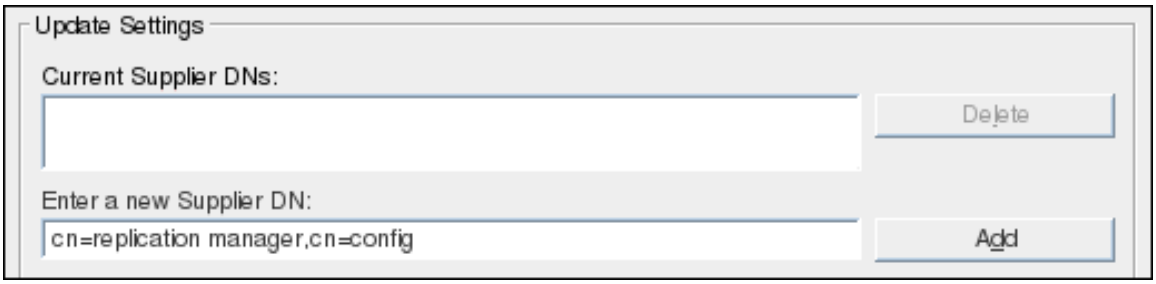
    4. In the **Replica Role** section, select the **Hub** radio button.

    5. In the **Common Settings** section, specify a purge delay in the **Purge delay** field.



        This option sets how often the state information stored for the replicated entries is purged.

    6. In the **Update Settings** section, specify the bind DN that the supplier will use to bind to the replica. Enter the supplier bind DN in the **Enter a new Supplier DN** field, and click **Add**. The supplier bind DN appears in the **Current Supplier DNs** list.



        The supplier bind DN should be the entry created in step 2. The supplier bind DN is a privileged user because it is not subject to access control in the replicated database.

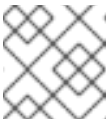
NOTE

There can be multiple supplier bind DNs per consumer but only one supplier DN per replication agreement.

7. Specify the URL for any supplier servers to which to refer updates.

   By default, all updates are first referred to the supplier servers that are specified here. If no suppliers are set here, updates are referred to the supplier servers that have a replication agreement that includes the current replica.

   Automatic referrals assume that clients bind over a regular connection; this has a URL in the form **ldap:**//*hostname*:*port*. For clients to bind to the supplier using TLS, use this field to specify a referral of the form **ldaps:**//*hostname*:*port*, where the **s** in **ldaps** indicates a secure connection.

   NOTE

   It is also possible to use IPv4 or IPv6 addresses instead of the host name.

5. Click **Save**.

When all the hubs are configured, then configure the supplier replica.

15.7.4. Setting up the Replication Agreements

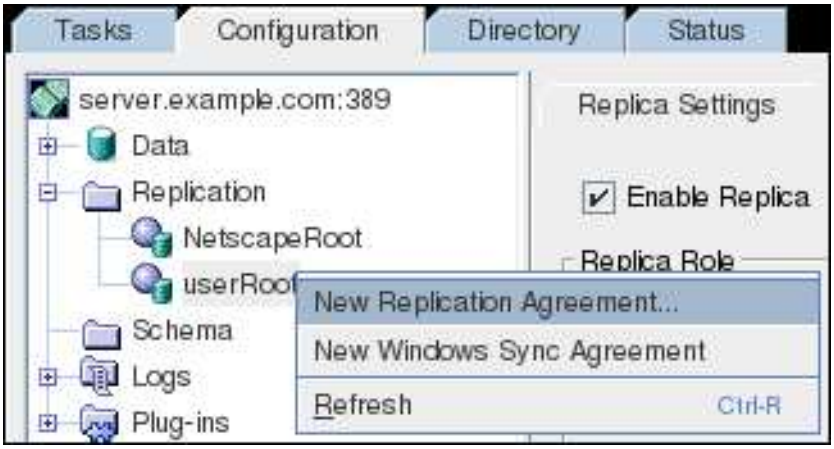
Cascading replication requires two sets of replication agreements, the first between the supplier and the hub and the second between the hub and the consumer. To set up the replication agreements:

1. Create the replication agreement on the supplier for the hub, then use the supplier server to initialize the replica on the hub server.

2. Then create the replication agreement on the hub for each consumer, and initialize the consumer replicas from the hub.

To set up a replication agreement:

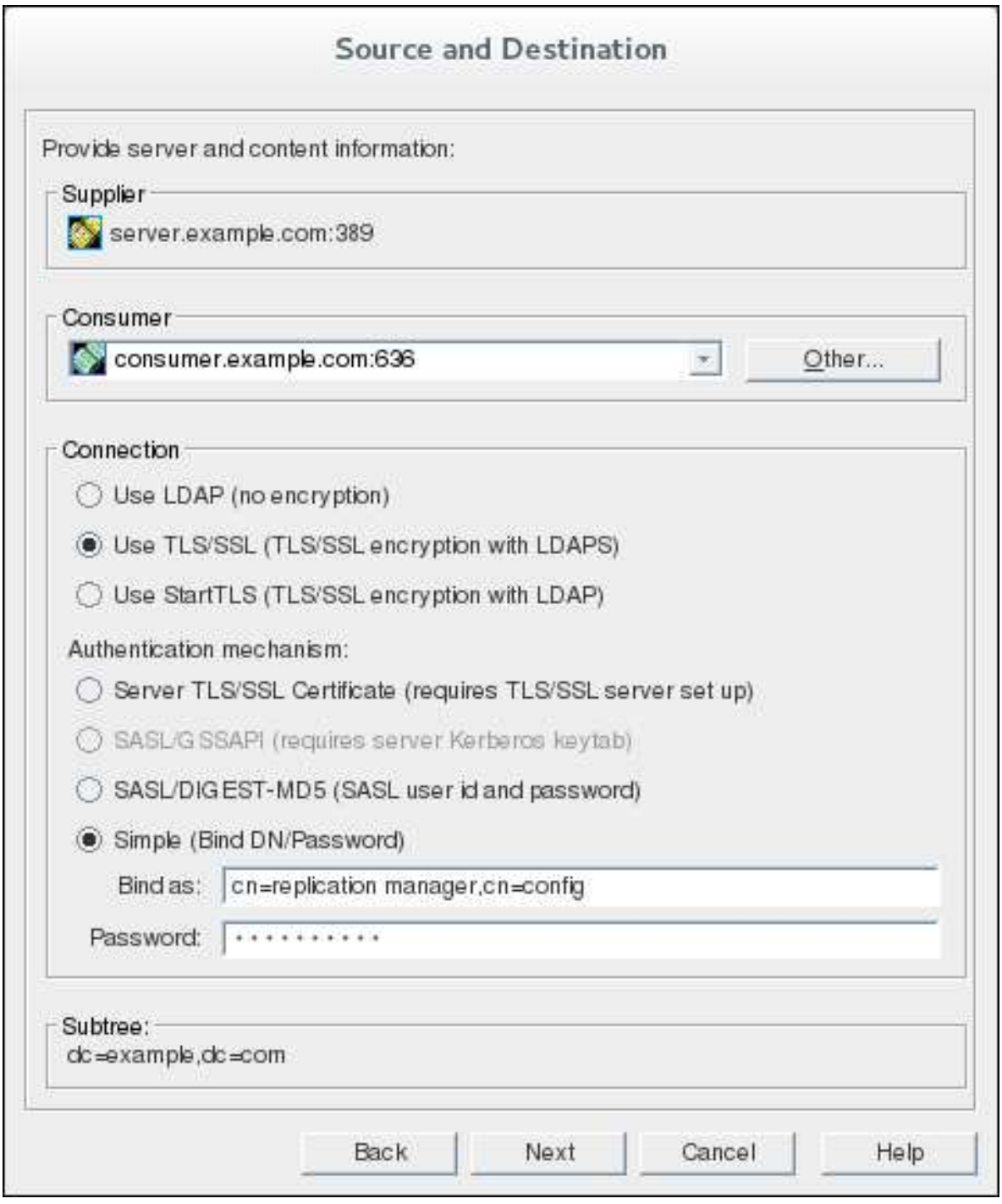
1. In the navigation tree of the **Configuration** tab, right-click the database to replicate, and select **New Replication Agreement**.

   Alternatively, highlight the database, and select **New Replication Agreement** from the **Object** menu to start the **Replication Agreement Wizard**.

2. In the first screen, fill in a name and description for the replication agreement, and hit **Next**.

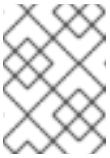
3. In the **Source and Destination** screen, fill in the URL (*hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses) for the consumer and the supplier bind DN and password on that consumer. If the target server is not available, hit in other to fill in the information manually.



- Unless there is more than one instance of Directory Server configured, by default, there are no consumers available in the drop-down menu. The server URL can be entered manually as *.hostname:port* or *IP_address:port*, with IPv4 or IPv6 addresses.

- The port listed is the non-TLS port, even if the Directory Server instance is configured to run over TLS. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.

- If TLS is enabled on the servers, it is possible to select the **Using encrypted SSL connection** radio button for TLS client authentication. Otherwise, fill in the supplier bind DN and password.



NOTE

If attribute encryption is enabled, a secure connection *must* be used for the encrypted attributes to be replicated.

4. Select the connection type. There are three options:

   - *Use LDAP*. This sets a standard, unencrypted connection.
   - *Use TLS/SSL*. This uses a secure connection over the server's secure LDAPS port, such as **636**. This setting is required to use TLS.
   - *Use Start TLS*. This uses Start TLS to establish a secure connection over the server's standard port.

NOTE

If secure binds are required for simple password authentication (Section 19.11.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (TLS and Start TLS connections or SASL authentication) is recommended, anyway.

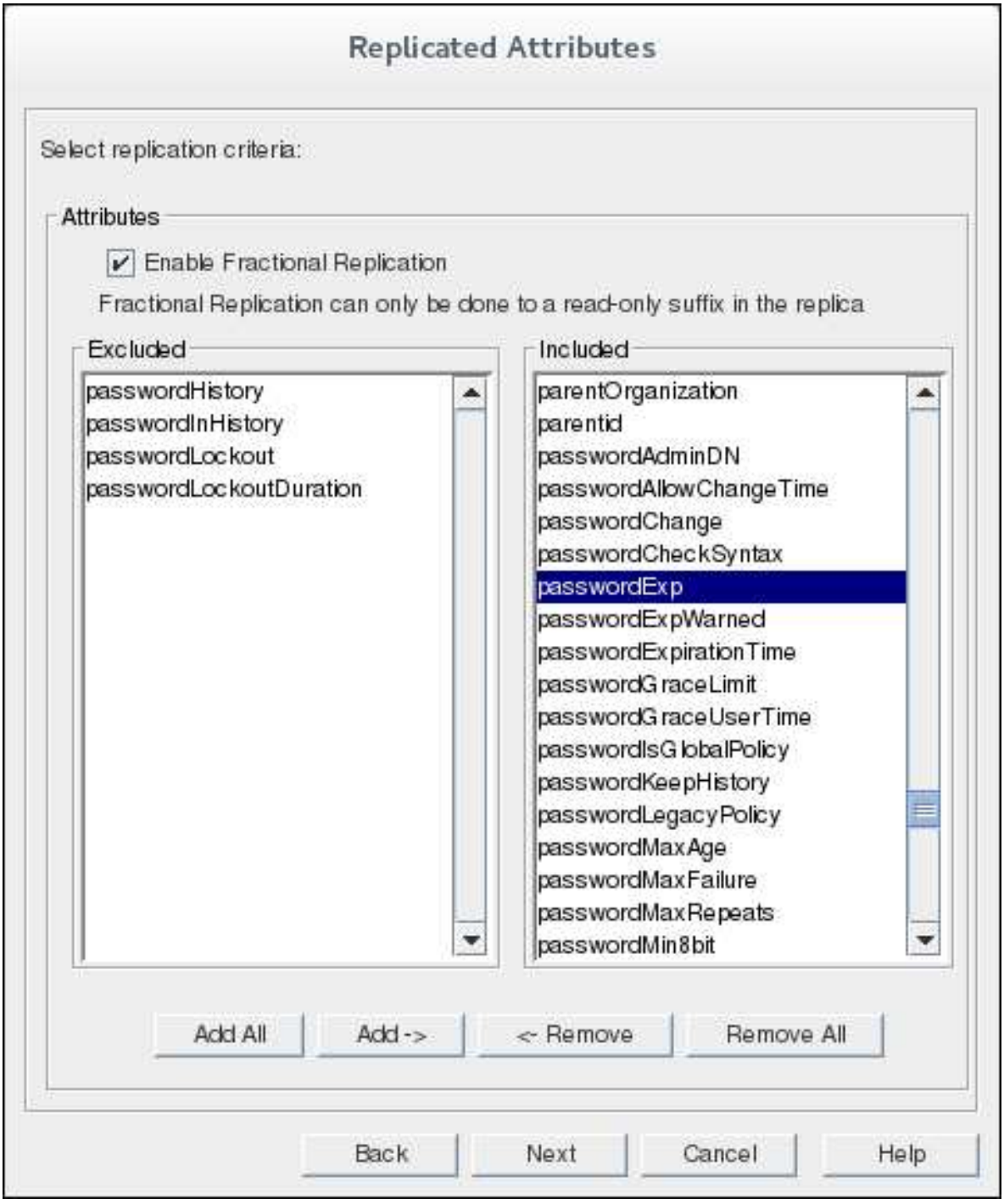
5. Select the appropriate authentication method and supply the required information. This gives the information that the supplier uses to authenticate and bind to the consumer server to send updates.

   - *Simple* means that the server connects over the standard port with no encryption. The only required information is the bind DN and password for the Replication Manager (which must exist on the consumer server).
   - *Server TLS/SSL Certificate* uses the supplier's TLS certificate to authenticate to the consumer server. A certificate must be installed on the supplier for certificate-based authentication, and the consumer server must have certificate mapping configured so that it can map the subject DN in the supplier's certificate to its Replication Manager entry.

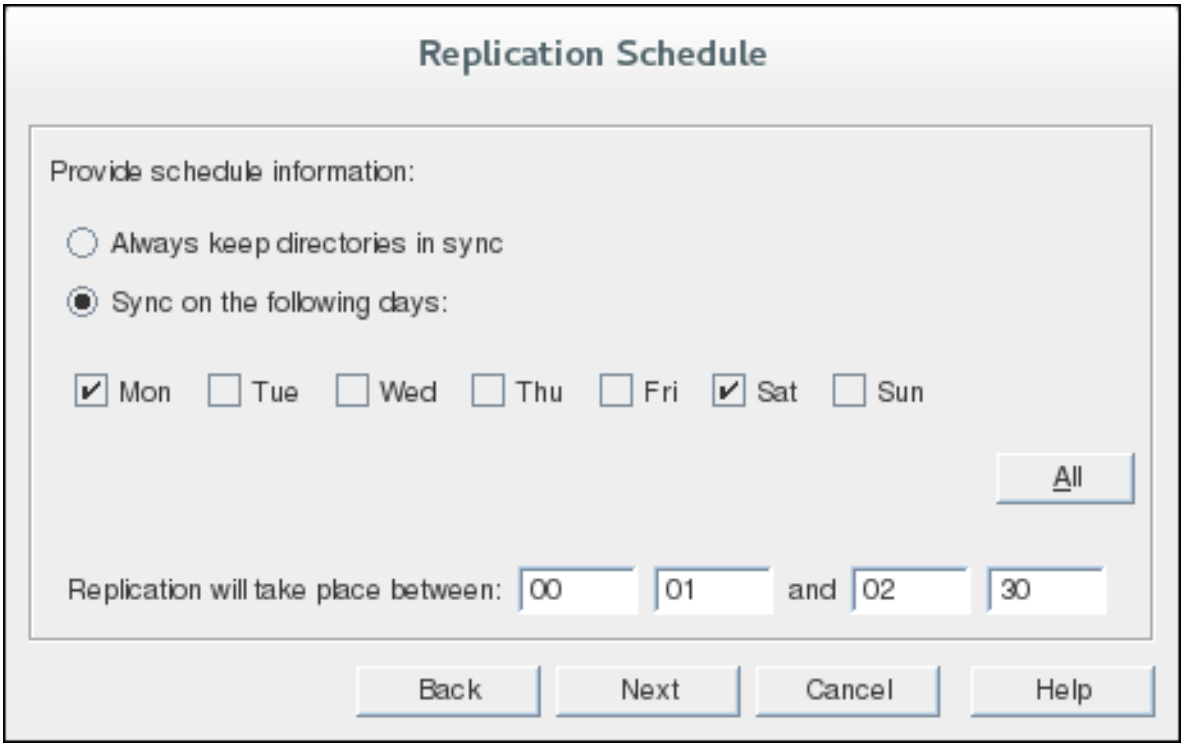
     Configuring TSL and certificate mapping is described in Section 9.4, "Enabling TLS".
   - *SASL/DIGEST-MD5*, like simple authentication, requires only the bind DN and password to authenticate. This can run over a standard or TLS connection.
   - *SASL/GSSAPI* requires the supplier server to have a Kerberos keytab (as in Section 9.10.2.2, "About the KDC Server and Keytabs"), and the consumer server to have a SASL mapping to map the supplier's principal to the real replication manager entry (as in Section 9.9.3.1, "Configuring SASL Identity Mapping from the Console").

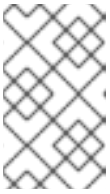
6. Hit **Next**.

7. Fractional replication controls which entry attributes are replicated between servers. By default, all attributes are replicated. To select attributes that will *not* be replicated to the consumer, check the **Enable Fractional Replication** check box. Then, highlight the attribute (or attributes) in the **Included** column on the right, and click **Remove**. All attributes that will not be replicated are listed in the **Excluded** column on the left, as well as in the summary the replication agreement is complete.



8. Set the schedule for when replication runs. By default, replication runs continually.





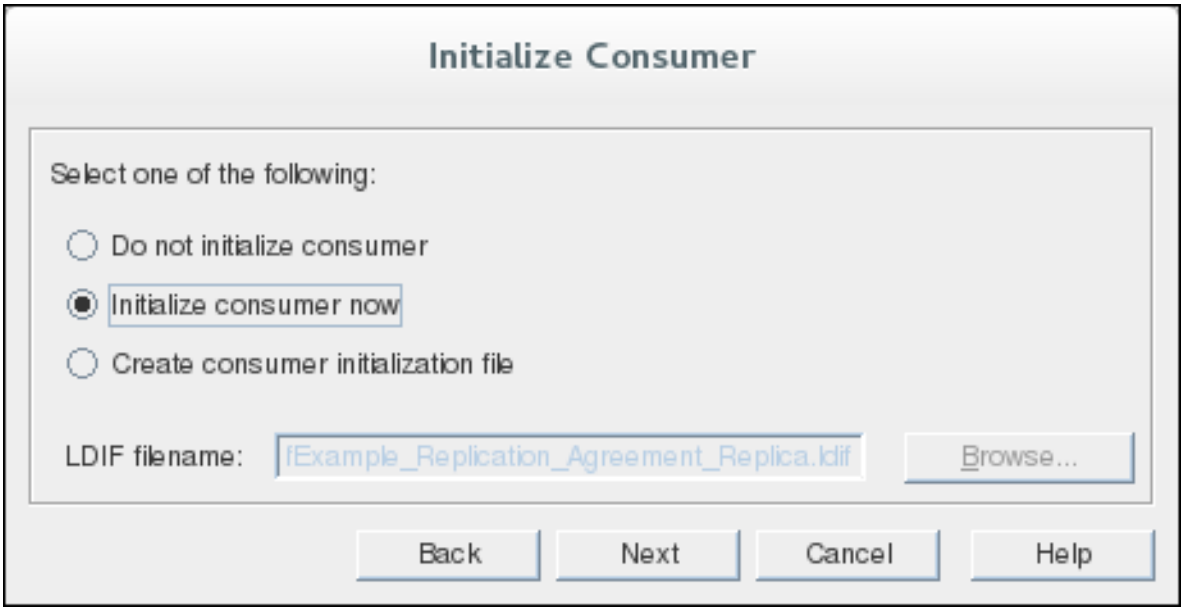
NOTE

The replication schedule cannot cross midnight (**0000**). So, it is possible to set a schedule that begins at **0001** and ends at **2359** on the same day, but it is not possible to set one that begins at **2359** on one day and ends at **0001** on the next.

Hit **Next**.

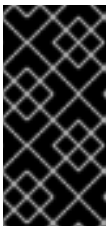
9. Set when the consumer is initialized. *Initializing* a consumer manually copies all data over from the supplier to the consumer. The default is to create an initialization file (an LDIF of all supplier data) so that the consumer can be initialized later. It is also possible to initialize the consumer as soon as the replication agreement is completed or not at all. For information on initializing consumers, see Section 15.18, "Initializing Consumers". For cascading replication, consider the following:

   - Create the supplier-hub replication agreement on the supplier first, and initialize the hub from the supplier.
   - Create the hub-consumer replication agreements on the hub, and initialize the consumers from the hub.



NOTE

Replication *will not* begin until the consumer is initialized.

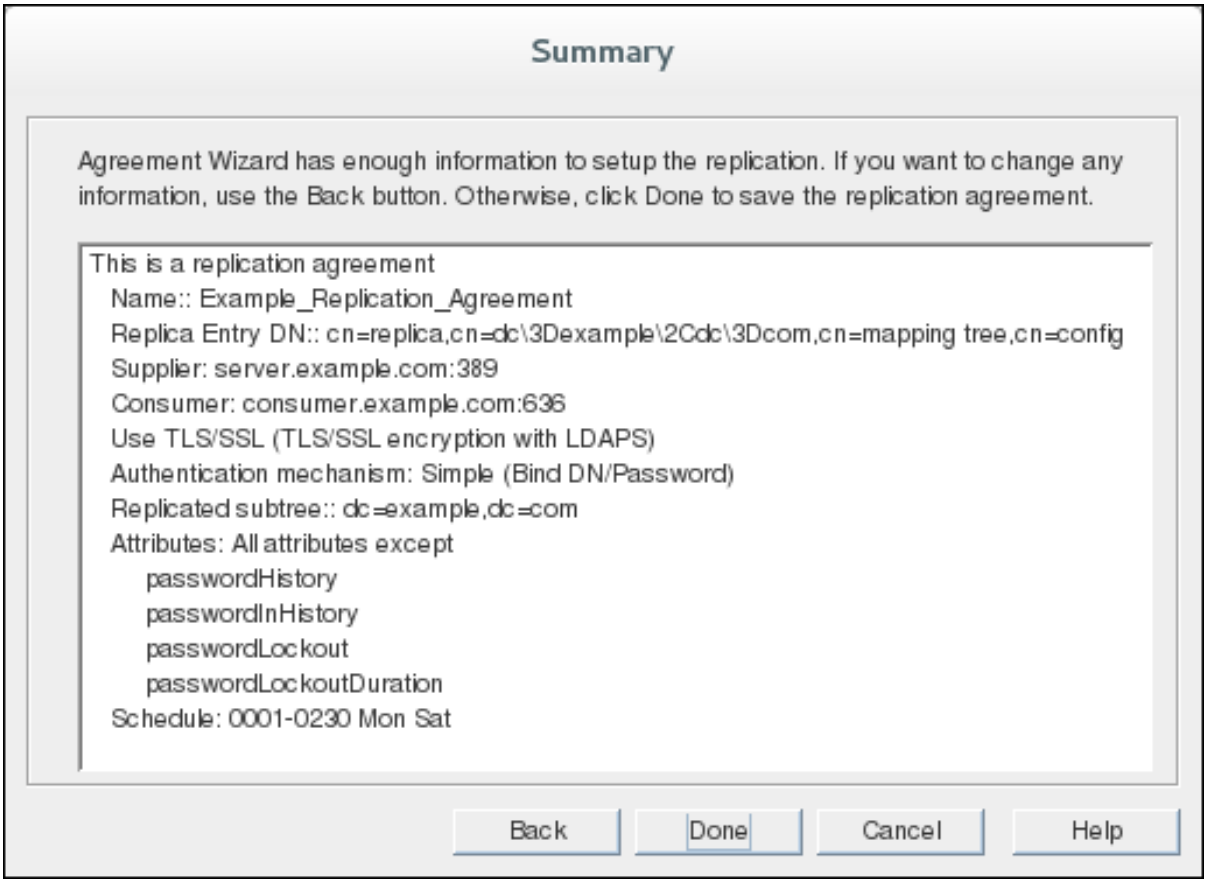


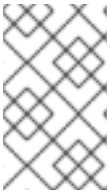
IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

Hit **Next**.

10. The final screen shows the settings for the replication agreement, as it will be included in the **dse.ldif** file. Hit **Done** to save the agreement.





NOTE

After creating a replication agreement, the connection type (TLS or non-TLS) cannot be change because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

15.8. TEMPORARILY SUSPENDING REPLICATION

To temporarily suspend replication, disable the replication agreement. When you re-enable the agreement, replication continues. For details, see Section 15.9, “Disabling and Re-enabling a Replication Agreement”.

15.9. DISABLING AND RE-ENABLING A REPLICATION AGREEMENT

To temporarily disable a replication agreement:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replication_agreement_name,cn=replica,cn=suffix_DN,cn=mapping tree,cn=config
changetype: modify
replace: nsds5ReplicaEnabled
nsds5ReplicaEnabled: off
```

To re-enable an replication agreement:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replication_agreement_name,cn=replica,cn=suffix_DN,cn=mapping tree,cn=config
changetype: modify
```

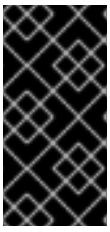
```
replace: nsds5ReplicaEnabled
nsds5ReplicaEnabled: on
```

15.10. MANAGING ATTRIBUTES WITHIN FRACTIONAL REPLICATION

As Section 15.1.7, "Replicating a Subset of Attributes with Fractional Replication" describes, fractional replication allows administrators to set attributes that are *excluded* from replication updates. Administrators can do this for a variety of performance reasons – to limit the number of large attributes that are sent over a network or to reduce the number of times that fixup tasks (like ***memberOf*** calculations) are run.

The list of attributes to exclude from replication are defined in the ***nsDS5ReplicatedAttributeList*** attribute. This attribute is part of the replication agreement and it can be configured in the replication agreement wizard in the Directory Server Console (or through the command line) when the replication agreement is created.

```
nsDS5ReplicatedAttributeList: (objectclass=*) $ EXCLUDE memberof authorityRevocationList
accountUnlockTime
```

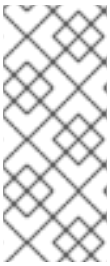


IMPORTANT

Directory Server requires the **(objectclass=*) $ EXCLUDE** part in the value of the ***nsDS5ReplicatedAttributeList*** attribute. If you edit the attribute directly, for example using the **ldapmodify** utility, you must specify this part together with the list of attributes as displayed in the example above.

15.10.1. Setting Different Fractional Replication Attributes for Total and Incremental Updates

When fractional replication is first configured, the list of excluded attributes applies to every update operation. Meaning, this list of attributes is excluded for a total update as well as regular incremental updates. However, there can be times when attributes should be excluded from incremental updates for performance but should be included in a total update to ensure the directory data sets are complete. In this case, it is possible to add a second attribute that defines a separate list of attributes to exclude from total updates, ***nsDS5ReplicatedAttributeListTotal***.



NOTE

nsDS5ReplicatedAttributeList is the primary fractional replication attribute. If only ***nsDS5ReplicatedAttributeList*** is set, then it applies to both incremental updates and total updates. If both ***nsDS5ReplicatedAttributeList*** and ***nsDS5ReplicatedAttributeListTotal*** are set, then ***nsDS5ReplicatedAttributeList*** only applies to incremental updates.

For example, every time a ***memberOf*** attribute is added to an entry, a memberOf fixup task is run to resolve the group membership. This can cause overhead on the server if that task is run every time replication occurs. Since a total update only occurs for a database which is newly-added to replication or that has been offline for a long time, running a memberOf fixup task after a total update is an acceptable option. In this case, the ***nsDS5ReplicatedAttributeList*** attribute lists ***memberOf*** so it is excluded from incremental updates, but ***nsDS5ReplicatedAttributeListTotal*** does not list ***memberOf*** so that it is included in total updates.

The exclusion list for incremental updates is set in the ***nsDS5ReplicatedAttributeList*** attribute for the replication agreement.

```
nsds5replicatedattributelist: (objectclass=*) $ EXCLUDE authorityRevocationList accountUnlockTime
memberof
```

If ***nsDS5ReplicatedAttributeList*** is the only attribute set, then that list applies to both incremental and total updates. To set a separate list for total updates, add the ***nsDS5ReplicatedAttributeListTotal*** attribute to the replication agreement.

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

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
add: nsDS5ReplicatedAttributeListTotal
nsDS5ReplicatedAttributeListTotal: (objectclass=*) $ EXCLUDE accountUnlockTime
```

NOTE

The ***nsDS5ReplicatedAttributeList*** attribute must be set for incremental updates before ***nsDS5ReplicatedAttributeListTotal*** can be set for total updates.

15.10.2. Preventing "Empty" Updates from Fractional Replication

Fractional replication allows a list of attributes which are removed from replication updates (***nsDS5ReplicatedAttributeList***). However, a changed to an excluded attribute still triggers a modify event and generates an empty replication update.

The ***nsds5ReplicaStripAttrs*** attribute adds a list of attributes which cannot be sent in an empty replication event and are stripped from the update sequence. This logically includes operational attribtes like ***modifiersName***.

For example, let's say that the ***accountUnlockTime*** attribute is excluded. John Smith's user account is locked and then the time period expires and it is automatically unlocked. Only the ***accountUnlockTime*** attribute has changed, and that attribute is excluded from replication. However, the operational attribute ***internalmodifytimestamp*** *also* changed. A replication event is triggered because John Smith's user account was modified – but the only data to send is the new modify time stamp and the update is otherwise emtpy. If there are a large number of attributes related to login times or password expiration times (for example), this could create a flood of empty replication updates that negatively affect server performance or that interfere with associated applications.

To prevent this, add the ***nsds5ReplicaStripAttrs*** attribute to the replication agreement to help tune the fractional replication behavior:

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

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
add: nsds5ReplicaStripAttrs
nsds5ReplicaStripAttrs: modifiersname modifytimestamp internalmodifiersname
internalmodifytimestamp
```

If a replication event is *not* empty, the stripped attributes *are* still replicated with the other changes. These attributes are removed from updates only if the event would otherwise be emtpy.

15.11. MAKING A READ-ONLY REPLICA UPDATABLE

Making a read-only server writable means changing the replica from a dedicated consumer or a hub to a supplier.

1. Make sure there are no updates in progress.
2. Stop the supplier server.
3. Enable the change log:

```
# ldapmodify -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/
```

4. Change the replica role:
   - To a single master:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
replace: nsDS5ReplicaType
nsDS5ReplicaType: 3
-
replace: nsDS5Flags
nsDS5Flags: 1
-
replace: nsDS5ReplicaId
nsDS5ReplicaId: unique_replica_id
-
delete: nsDS5ReplicaBindDN
```

   - To a multi master:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: modify
replace: nsDS5ReplicaType
nsDS5ReplicaType: 3
-
replace: nsDS5Flags
nsDS5Flags: 1
-
replace: nsDS5ReplicaId
nsDS5ReplicaId: unique_replica_id
```

```
-
replace: nsDS5ReplicaBindDN
nsDS5ReplicaBindDN: cn=Replication Manager,cn=config
```

5. Stop the Directory Server instance:

```
# systemctl stop dirsrv
```

6. Back up the **/etc/dirsrv/slapd-*instance*/dse.ldif** file:

```
# cp /etc/dirsrv/slapd-instance_name/dse.ldif \
    /etc/dirsrv/slapd-instance_name/dse.ldif-1
```

   Do not name the backup file **dse.ldif.bak**. Directory Server uses this file name to keep a known working copy of the **dse.ldif** file.

7. Edit the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file.

   a. Search for replication agreements. For example:

```
dn: cn=replica,cn=dc\5c3Dexample\5c2Cdc\5c3Dcom,cn=mapping tree,cn=config
```

   b. Remove the line containing the ***nsState*** attribute from every replication agreement.

8. Start the Directory Server instance:

```
# systemctl start dirsrv.target
```

9. Monitor the error log file for error messages. For details, see Section E.2.3, “Viewing Logs” .

   If the replication fails:

   a. Delete all replication agreements: Section 15.12, “Removing a Supplier from the Replication Topology”.

   b. Disable replication: Section 15.9, “Disabling and Re-enabling a Replication Agreement” .

   c. Remove the changelog configuration: Section 15.15, “Removing the Changelog”.

   d. Restart the Directory Server and Admin Console:

```
# systemctl restart dirsrv.target
# systemctl restart dirsrv-admin.service
```

   e. Enable replication: Section 15.9, “Disabling and Re-enabling a Replication Agreement” .

   f. Create replication agreements: Section 15.3, “Replication Scenarios”.

15.12. REMOVING A SUPPLIER FROM THE REPLICATION TOPOLOGY

Removing a supplier cleanly from the replication topology is more complex than simply removing the supplier entry. This is because every supplier in the topology stores information about other suppliers, and they retain that information even if a supplier suddenly becomes unavailable.

Information about the replication topology, that is all suppliers which supply updates to each other and other replicas within the same replication group, is contained in a set of metadata called the *replica update vector (RUV)*. The RUV contains information about the supplier such as its ID and URL, its latest change state number (CSN) on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

To remove a supplier cleanly, its metadata must be removed along with the configuration entries.

1. *On the replica to remove* , put the database into read-only mode to prevent any updates.

```
# ldapmodify -D "cn=Directory Manager" -W -x -p 389 -h dead-replica.example.com

dn: cn=userRoot,cn=ldbm database,cn=plugins,cn=config
changetype: modify
replace: nsslapd-readonly
nsslapd-readonly: on
```

   Allow a few minutes for the replica to flush all of its pending changes.

2. *On all other suppliers in the topology* , delete the replication agreement with the replica to be removed.

```
# ldapmodify -D "cn=Directory Manager" -W -x -p 389 -h replica1.example.com

dn: cn=Agmt_with_dead-replica,cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
tree,cn=config
changetype: delete
```

3. *On the replica to remove* , get the replica ID for the replica to remove. This is in the ***nsds5replicaid*** attribute in the configuration entry.

```
# ldapsearch -xLLL -D "cn=Directory Manager" -W -s sub -b cn=config
objectclass=nsds5replica nsds5replicaid

dn: cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsds5replicaid: 55
...
```

4. *On the replica to remove* , remove all replication agreement entries and its own configuration entry.

```
# ldapmodify -D "cn=Directory Manager" -W -x -p 389 -h dead-replica.example.com

dn: cn=to_replica1,cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
tree,cn=config
changetype: delete
...

dn: cn=to_replica2,cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping
tree,cn=config
changetype: delete

dn: cn=dead-replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
changetype: delete
```

5. *On one of the other master servers in the topology* , run the **clean** command on the replica ID:

```
# ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=clean 55, cn=cleanallruv, cn=tasks, cn=config
objectclass: extensibleObject
replica-base-dn: dc=example,dc=com
replica-id: 55
cn: clean 55
```

It is possible to monitor the progress of the task on the other replicas by searching on the tombstone entry on each replica:

```
# ldapsearch -xLLL -D "cn=Directory Manager" -W -h remaining-replica.example.com -b
"dc=example,dc=com" '(&(nsuniqueid=ffffffff-ffffffff-ffffffff-ffffffff)(objectclass=nstombstone))'
nsds50ruv
```

15.13. MANAGING DELETED ENTRIES WITH REPLICATION

When an entry is deleted, it is not immediately removed from the database. Rather, it is converted into a *tombstone* entry, a kind of backup entry that is used by servers in replication to resolve conflicts. The tombstone entry preserves state information about the original entry.

If there is ever a replication conflict, then the supplier uses the replica ID (the server where the change was initiated) and the timestamp of the change in the change sequence number to resolve the conflict. The oldest change wins. As with deleted entries, deleted attributes are also kept in tombstone entries.

Tombstones are not preserved indefinitely. A purge job is run periodically, at a specified interval (set in the ***nsDS5ReplicaTombstonePurgeInterval*** attribute); the purge removes old tombstone entries. Tombstone entries are saved for a given amount of time (set in the ***nsDS5ReplicaPurgeDelay*** attribute); once a tombstone entry is older than the delay period, it is reaped at the next purge job.

Both the purge delay and the purge interval are set on the replica entry for a supplier server in the **cn=replica,cn=***replicated suffix***,cn=mapping tree,cn=config** configuration entry. There are two considerations when defining the purge settings for replication:

- The purge operation is time-consuming, especially if the server handles a lot of delete operations. Do not set the purge interval too low or it could consume too many server resources and affect performance.

- Suppliers use change information, including tombstone entries, to prime replication after initialization. There should be enough of a backlog of changes to effectively re-initialize consumers and to resolve replication conflicts. Do not set the purge delay (the age of tombstone entries) too low or you could lose information required to resolve replication conflicts.

  Set the purge delay so that it is slightly longer than the longest replication schedule in the replication topology. For example, if the longest replication interval is 24 hours, keep tombstone entries around for 25 hours. This ensures that there is enough change history to initialize consumers and prevent the data stored in different suppliers from diverging.

To change the purge settings:

```
# ldapmodify -D "cn=Directory Manager" -W -x  -h supplier1.example.com
```

```
dn: cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
replace: nsDS5ReplicaTombstonePurgeInterval
nsDS5ReplicaTombstonePurgeInterval: 43200 # in seconds, 12 hours
-
changetype: modify
replace: nsDS5ReplicaPurgeDelay
nsDS5ReplicaPurgeDelay: 90000 # in seconds, 25 hours
```

NOTE

To clean up the tombstone entries and the state information immediately, set a very small value to the ***nsDS5ReplicaTombstonePurgeInterval*** and ***nsDS5ReplicaPurgeDelay*** attributes. Both attributes have values set in seconds, so the purge operations can be initiated almost immediately.

WARNING

Always use the purge intervals to clean out tombstone entries from the changelog. **Never delete tombstone entries manually.**

15.14. CONFIGURING CHANGELOG ENCRYPTION

To increase security, Directory Server supports encrypting the changelog. This section explains how to enable this feature.

Prerequisites

The server must have a certificate and key stored in the network security services (NSS) database. Therefor, enable TLS encryption on the server as described in Section 9.4.1, "Enabling TLS in Directory Server".

Procedure

To enable changelog encryption:

1. Except for the server on which you want to enable changelog encryption, stop all instances in the replication topology by entering the following command:

   ```
   # systemctl stop dirsrv@instance_name
   ```

2. On the server where you want to enable changelog encryption:

   a. Create a task that exports the changelog:

   ```
   # ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
   dn: cn=replica,cn=suffix,cn=mapping tree,cn=config
   changetype: modify
   add: nsds5Task
   nsds5Task: CL2LDIF
   ```

Directory Server stores the export in the **/var/lib/dirsrv/slapd-*instance_name*/changelogdb/** directory.

b. Stop the instance:

```
# systemctl stop dirsrv@instance_name
```

c. Add the following setting to the **dn: cn=changelog5,cn=config** entry in the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file:

```
nsslapd-encryptionalgorithm: AES
```

d. Start the instance:

```
# systemctl start dirsrv@instance_name
```

e. Create a task to import the changelog:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replica,cn=suffix,cn=mapping tree,cn=config
changetype: modify
add: nsds5Task
nsds5Task: LDIF2CL
```

3. Start all instances on the other servers in the replication topology using the following command:

```
# systemctl start dirsrv@instance_name
```

Verification

To verify that the changelog is encrypted, run the following steps on the server with the encrypted changelog:

1. Make a change in the LDAP directory, such as updating an entry.
2. Stop the instance:

```
# systemctl stop dirsrv@instance_name
```

3. Enter the following command to display parts of the changelog:

```
# dbscan -f /var/lib/dirsrv/slapd-instance_name/changelogdb/replica_name_replGen.db | tail -
50
```

If the changelog is encrypted, you see only encrypted data.

4. Start the instance:

```
# systemctl start dirsrv@instance_name
```

15.15. REMOVING THE CHANGELOG

The changelog is a record of all modifications on a given replica that the supplier uses to replay these modifications to replicas on consumer servers (or suppliers in the case of multi-master replication).

If a supplier server goes offline, it is important to be able to delete the changelog because it no longer holds a true record of all modifications and, as a result, should not be used as a basis for replication. A changelog can be effectively deleted by deleting the log file.

15.15.1. Removing the Changelog using the Command Line

To remove the changelog from the supplier server:

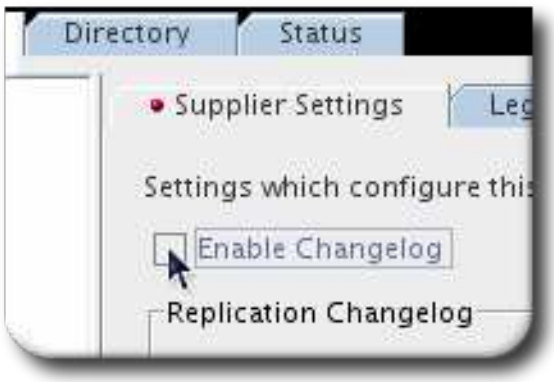
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=changelog5,cn=config
changetype: delete
```

Directory Server automatically removes the content in the changelog directory after you removed the **cn=changelog5,cn=config** entry.

15.15.2. Removing the Changelog using the Console

To remove the changelog from the supplier server:

1. In the Directory Server Console, select the **Configuration** tab.
2. Select the **Replication** folder in the left navigation tree and then the **Supplier Server Settings** tab in the right pane.
3. Clear the **Enable Changelog** check box.



4. Click **Save**.
5. Restart Directory Server. See Section 1.4.2, "Starting and Stopping a Directory Server Instance Using the Console".
6. Reinitialize the consumers. See Section 15.18, "Initializing Consumers".

15.16. MOVING THE REPLICATION CHANGELOG DIRECTORY

In certain situations, you might want to change the Directory Server replication changelog directory. For example, to change the directory to **/var/lib/dirsrv/slapd-*instance_name*/new_changelogdb/**:

1. Display the current directory:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
    -b "cn=changelog5,cn=config" nsslapd-changelogdir
...
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/changelogdb/
```

You need the displayed path in a later step to move the directory.

2. Set the new path:

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

dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogdir
nsslapd-changelogdir: /var/lib/dirsrv/slapd-instance_name/new_changelogdb/
```

3. Stop the Directory Server instance:

```
# systemctl stop dirsrv@instance_name
```

4. Move the content of the previous directory to **/var/lib/dirsrv/slapd-*instance_name*/new_changelogdb/**:

```
# mv /var/lib/dirsrv/slapd-instance_name/changelogdb/ \
    /var/lib/dirsrv/slapd-instance_name/new_changelogdb/
```

5. Delete the previous directory:

```
# rm /var/lib/dirsrv/slapd-instance_name/changelogdb/
```

6. Start the Directory Server instance:

```
# systemctl start dirsrv@instance_name
```

15.17. TRIMMING THE REPLICATION CHANGELOG

The Directory Server changelog manages a list of received and processed changes. It includes changes of clients ran on the server and additionally directory changes from other replication partners. Using the default settings, Directory Server does not automatically remove entries and the changelog grows infinitely. To control which entries are removed, use the following parameters:

- ***nsslapd-changelogmaxage*** (recommended): Removes entries if they exceed the time set in this parameter.
- ***nsslapd-changelogmaxentries***: Removes the oldest entries if the total number of records exceed the value set in this parameter.

Any record and all subsequently created records remains in the changelog until it is successfully replicated to all servers in the topology. For example, this occurs in a situation when a Directory Server master was removed from the topology, but the replica update vector (RUV) had not been removed.

15.17.1. Enabling Replication Changelog Trimming

To enable replication changelog trimming and automatically remove entries that are older than 7 days (**7d**):

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

dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 7d
```

NOTE

Red Hat recommends setting a maximum age in ***nsslapd-changelogmaxage*** instead of a maximum number of entries in ***nsslapd-changelogmaxentries***. The time set in ***nsslapd-changelogmaxage*** should match the replication purge delay set in ***nsDS5ReplicaPurgeDelay***. For details about ***nsDS5ReplicaPurgeDelay***, see the parameter description in the *Red Hat Directory Server Configuration, Command, and File Reference*.

15.17.2. Manually Reducing the Size of a Large Changelog

When trimming the replication changelog was not enabled and the database grew to a large size, reduce the changelog size manually in the short term:

1. To be able to reset the parameters after reducing the changelog size, display the current values of corresponding parameters. For example:

   ```
   # ldapsearch -x -D 'cn=Directory Manager' -W -b "cn=changelog5,cn=config" \
       nsslapd-changelogmaxage nsslapd-changelogcompactdb-interval \
       nsslapd-changelogtrim-interval nsslapd-changelogmaxage

   dn: cn=changelog5,cn=config
   nsslapd-changelogmaxage: 7d
   nsslapd-changelogcompactdb-interval: 2592000
   nsslapd-changelogtrim-interval: 300
   ```

   Parameters that are not displayed in the output are not set and Directory Server uses their default values. For the default values, see the parameter descriptions in the *Red Hat Directory Server Configuration, Command, and File Reference* .

2. Reduce the following parameter's value temporarily:

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

   dn: cn=changelog5,cn=config
   changetype: modify
   replace: nsslapd-changelogmaxage
   nsslapd-changelogmaxage: 3d
   -
   replace: nsslapd-changelogtrim-interval
   nsslapd-changelogtrim-interval: 30
   -
   replace: nsslapd-changelogcompactdb-interval
   nsslapd-changelogcompactdb-interval: 300
   ```

Using these settings, Directory Server removes changelog entries older than 3 days (***nsslapd-changelogmaxage***) within the next 30 seconds (***nsslapd-changelogtrim-interval***).

3. Restart the Directory Server instance so that the new value of the ***nsslapd-changelogcompactdb-interval*** parameter takes effect:

```
# systemctl restart dirsrv@instance
```

After the next database update, the database is automatically compacted within the time interval set in the ***nsslapd-changelogcompactdb-interval*** parameter.

4. Reset the updated parameters to their previous values. For example:

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

dn: cn=changelog5,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 7d
-
replace: nsslapd-changelogtrim-interval
nsslapd-changelogtrim-interval: 300
-
replace: nsslapd-changelogcompactdb-interval
nsslapd-changelogcompactdb-interval: 2592000
```

IMPORTANT

For performance reasons, do not permanently use too short interval settings.

5. Restart the Directory Server instance:

```
# systemctl restart dirsrv@instance
```

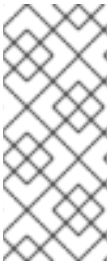
15.18. INITIALIZING CONSUMERS

Once a replication agreement is created, the consumer must be *initialized*; that is, the data must be physically copied from the supplier server to the consumer servers.

NOTE

Replication *will not* begin until the consumer is initialized.

- Section 15.18.1, “When to Initialize a Consumer”
- Section 15.18.2, “Online Consumer Initialization Using the Console”
- Section 15.18.3, “Initializing Consumers Online Using the Command Line”
- Section 15.18.4, “Manual Consumer Initialization Using the Command Line”



NOTE

For large databases, the ***nsslapd-idletimeout*** setting must be set to a large enough time period (or even an unlimited period) to allow the entire database to be initialized before the operation times out. Alternatively, the ***nsIdleTimeout*** setting for the supplier bind DN entry can be set high enough to allow the online initialization operation to complete, without having to change the global setting.

15.18.1. When to Initialize a Consumer

Consumer initialization involves copying data from the supplier server to the consumer server. Once the subtree has been physically placed on the consumer, the supplier server can begin replaying update operations to the consumer server.

Under normal operations, the consumer should not ever have to be reinitialized. However, any time there is a chance that there is a big discrepancy between the supplier's data and the consumer's, reinitialize the consumer. For example, if the data on the supplier server is restored from backup, then all consumers supplied by that server should be reinitialize. As another example, if the supplier has not been able to contact the consumer for a long time, like a week, the supplier may determine that the consumer is too far out of date to be updated, and must be reinitialized.

The consumer can either be initialized online using the Console or manually using the command line. Online consumer initialization using the Console is an effective method of initializing a small number of consumers. However, since each replica is initialized in sequence, this method is not suited to initializing a large number of replicas. Online consumer initialization is the method to use when the consumer is initialized as part of configuring the replication agreement on the supplier server.

Manual consumer initialization using the command line is a more effective method of initializing a large number of consumers from a single LDIF file.

15.18.2. Online Consumer Initialization Using the Console

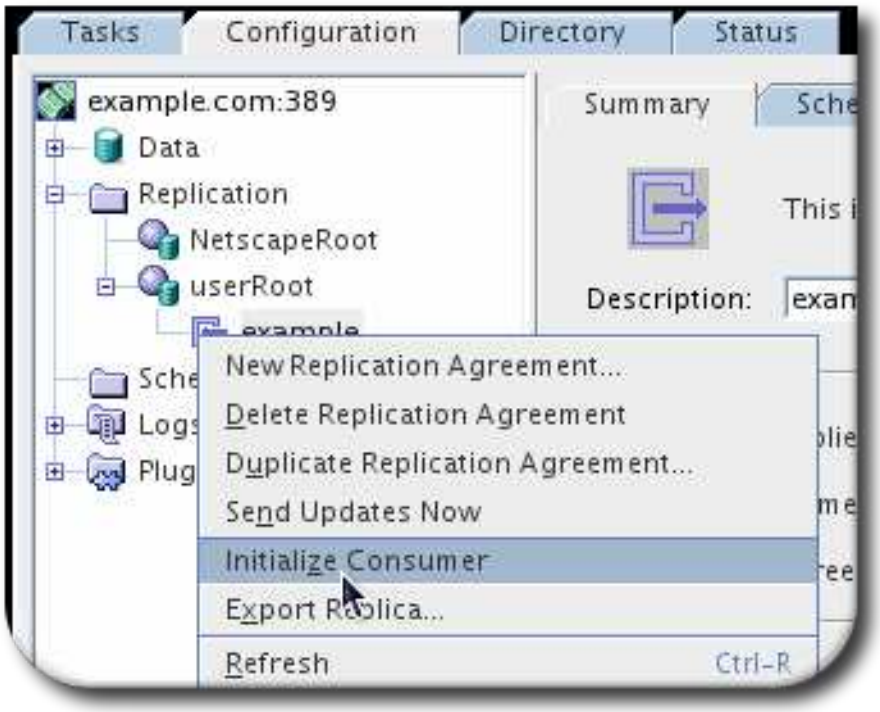
Online consumer initialization using the Console is the easiest way to initialize or reinitialize a consumer. However, for replicating across a slow link, this process can be very time-consuming, and manual consumer initialization using the command line may be a more efficient approach. This is described in more detail Section 15.18.4, “Manual Consumer Initialization Using the Command Line” .

NOTE

When a consumer server is being initialized using the online consumer creation method, all operations (including searches) on the replica are referred to the supplier server until the initialization process is completed.

To initialize or reinitialize a consumer online:

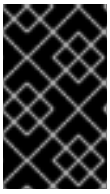
1. Create a replication agreement.
2. On the supplier server, on the Directory Server Console, select the **Configuration** tab.
3. Expand the **Replication** folder, then expand the replicated database. Right-click the replication agreement, and choose **Initialize Consumer** from the pop-up menu.



A message opens warning that any information already stored in the replica on the consumer will be removed.

4. Click **Yes** in the confirmation box.

Online consumer initialization begins immediately. To check the status of the online consumer initialization, open the **Summary** tab in the **Status** box. If online consumer initialization is in progress, the status shows that a replica is being initialized.



IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

To update this window, right-click the replicated database icon in the navigation tree, and choose **Refresh Replication Agreements**. When online consumer initialization finishes, the status changes to reflect this.

For more information about monitoring replication and initialization status, see Section 15.24, “Monitoring Replication Status”.

15.18.3. Initializing Consumers Online Using the Command Line

Online consumer initialization can be performed through the command line by adding the ***nsds5BeginReplicaRefresh*** attribute to the replication agreement entry. This attribute is absent by default, and it will be automatically deleted once the consumer initialization is complete.

1. Find the DN of the replication agreement on the supplier server that is for the consumer to be initialized. For example:

```
# ldapsearch -h supplier1.example.com -p 389 -D "cn=Directory Manager" -W -s sub
    -b cn=config "(objectclass=nsds5ReplicationAgreement)"
```

This command returns all of the replication agreements configured on the supplier in LDIF format. Get the DN of the replication agreement with the consumer to be initialized. This is the replication agreement which will be edited.

2. Edit the replication agreement, and add the ***nsds5BeginReplicaRefresh*** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -x -h supplier1.example.com

dn: cn=ExampleAgreement,cn=replica,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config
changetype: modify
replace: nsds5BeginReplicaRefresh
nsds5BeginReplicaRefresh: start
```

ldapmodify does not prompt for input; simply type in the LDIF statement, and then hit enter twice when the LDIF statement is complete. Close the **ldapmodify** utility by hitting **Ctrl+C**.

To check the initialization status, do an **ldapsearch** for the replication agreement entry.

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x -s base -b
'cn=ExampleAgreement,cn=dc\=example\,dc\=com,cn=mapping tree,cn=config' '(objectclass=*)'
```

If the ***nsds5BeginReplicaRefresh*** attribute is present, the initialization is still in progress. If the initialization is complete, then the attribute ***nsds5ReplicaLastInitStatus*** shows the status. If the initialization was successful, the value of ***nsds5ReplicaLastInitStatus*** is **Total update succeeded**. If the initialization was not successful, this attribute shows information about the error; check the error logs for both the supplier and consumer for additional information.

IMPORTANT

For multi-master replication, be sure that consumers are only initialized **once**, by one supplier. When checking the replication status, be sure to check the replication agreement entry, on the appropriate supplier, which was used to initialize the consumer.

The replication monitoring attributes are described in more detail in the *Red Hat Directory Server Configuration, Command, and File Reference*.

15.18.4. Manual Consumer Initialization Using the Command Line

Manual consumer initialization using the command line is the fastest method of consumer initialization for sites that are replicating very large numbers of entries. However, the manual consumer initialization process is more complex than the online consumer initialization process. Red Hat suggests using the manual process whenever the online process is inappropriate due to performance concerns.

Initializing or reinitializing a server manually has three steps:

1. Create a replication agreement.
2. Export the replica on the supplier server to an LDIF file.

   See Section 15.18.4.1, “Exporting a Replica to LDIF” .
3. Import the LDIF file with the supplier replica contents to the consumer server.

   See Section 15.18.4.2, “Importing the LDIF File to the Consumer Server” .

15.18.4.1. Exporting a Replica to LDIF

There are three ways to convert a replica database to LDIF:

- When creating a replication agreement, by selecting **Create consumer initialization file** in the **Initialize Consumer** dialog box of the **Replication Agreement Wizard**.
- From the Directory Server Console, by right-clicking the replication agreement under the **Replication** folder and choosing **Create LDIF File** from the pop-up menu.
- From the command line by using the export command, as described in Section 6.2.3, "Exporting a Database to LDIF Using the Command Line". Exporting to LDIF with any of the command-line tools requires using an option to export the database as a replica; this means that the exported LDIF contains the proper entries to initialize the consumer when the LDIF is imported.

  For the **db2ldif** and **db2ldif.pl** scripts, this is the **-r** option. For example:

  ```
  # db2ldif -r -n database1 -a /export/output.ldif
  ```

  For the **cn=export,cn=tasks,cn=config** entry, this is the ***nsExportReplica*** attribute.

  ```
  #ldapmodify -a -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

  dn: cn=example export,cn=export,cn=tasks,cn=config
  changetype: add
  objectclass: extensibleObject
  cn: example export
  nsInstance: userRoot
  nsFilename: /home/files/example.ldif
  nsExportReplica: true
  ```

15.18.4.2. Importing the LDIF File to the Consumer Server

Import the LDIF file which contains the supplier replica contents to the consumer server by using the import features in the Directory Server Console or by using either the **ldif2db** script or **ldif2db.pl** script. Both import methods are described in Section 6.1.4, "Importing from the Command Line".

NOTE

With the **ldif2db.pl** script, the LDIF file import operation does not require a server restart. For more information on command-line scripts, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

15.19. FORCING REPLICATION UPDATES

When a Directory Server involved in replication is stopped for regular maintenance, it must be updated immediately when it comes back online. In the case of a supplier in a multi-master environment, the directory information needs to be updated by the other supplier in the multi-master set. In other cases, when a hub or a dedicated consumer is taken offline for maintenance, when they come back online, they need to be updated by the supplier server.

Even if the replication agreements are configured to keep the supplier and consumer servers always in sync, it is not sufficient to bring back up-to-date a server that has been offline for over five minutes. The **Always Keep in Sync** option means that the server generates a replication operation for every update

operation it processes. However, if this replication operation cannot be performed because the consumer is offline, the operation times out after 10 minutes.

NOTE

The procedures described in this section can only be used when replication is already set up and consumers have been initialized.

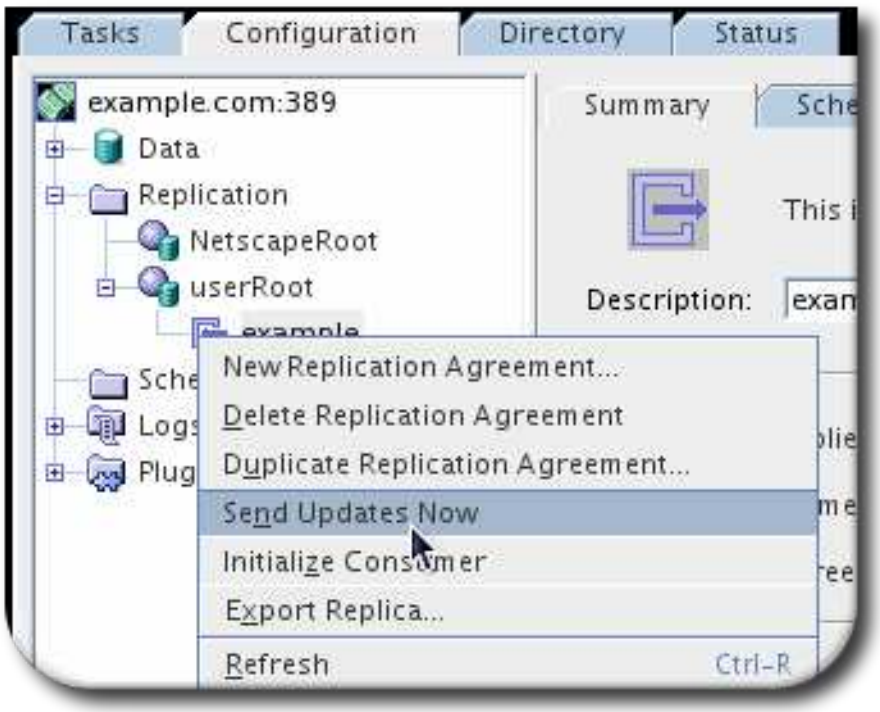
To ensure that directory information will be synchronized immediately when a server comes back online, use either the Directory Server Console on the supplier server that holds the reference copy of the directory information or a customizable script.

15.19.1. Forcing Replication Updates from the Console

To ensure that replication updates are sent immediately when a consumer or a supplier in a multi-master replication configuration comes back online after a period of time, do the following on the supplier server that holds the most recent version of the directory information:

1. In the Directory Server Console, click the **Configuration** tab, expand the **Replication** folder and database nodes, and select the replication agreement corresponding to the replica to update.

2. Right click the replication agreement, and choose **Send Updates Now** from the drop-down list.



This initiates replication toward the server that holds the information that needs to be updated.

15.19.2. Forcing Replication Updates from the Command Line

To force replication updates, disable and re-enable the replication agreement. For details, see Section 15.9, "Disabling and Re-enabling a Replication Agreement" .

15.20. REPLICATION OVER TLS

For security reasons, the Directory Servers involved in replication should be configured so that all replication operations occur over an TLS connection. To use replication over TLS:

- Configure both the supplier and consumer servers to use TLS.
- Configure the consumer server to recognize the supplier server's certificate as the supplier DN. Do this only to use TLS client authentication rather than simple authentication.

These procedures are described in Section 9.4, "Enabling TLS".

If attribute encryption is enabled, a secure connection is required for replication.

NOTE

Replication configured over TLS with certificate-based authentication will fail if the supplier's certificate is only capable of behaving as a server certificate, and not also a client during an TLS handshake. Replication with certificate-based authentication uses the Directory Server's server certificate for authentication to the remote server.

If you use **certutil** to generate the Certificate Signing Request (CSR), pass the **--*nsCertType=sslClient,sslServer*** option to the command to set the certificate required type.

When the servers are configured to use TLS, configure an TLS connection for replication in the **Replication Agreement Wizard**. The **Source and Destination** sets how to bind between the supplier and the consumer, and this is where TLS is set.

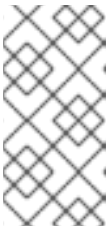
There are two ways to use TLS for replication:

- Select **SSL Client Authentication**.

  With TLS client authentication, the supplier and consumer servers use certificates to authenticate to each other.

- Select **Simple Authentication**.

  With simple authentication, the supplier and consumer servers use a bind DN and password to authenticate to each other, which are supplied in the **Replication Agreement Wizard** text fields provided. Simple authentication takes place over a secure channel but without certificates.



NOTE

If secure binds are required for simple password authentication (Section 19.11.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (TLS and Start TLS connections or SASL authentication) is recommended, anyway.

Once a replication agreement is created, the connection type (TLS or non-TLS) cannot be changed in the agreement because LDAP and LDAPS connections use different ports. To change the connection type, re-create the replication agreement.

Also, the port listed for the consumer is the non-TLS port, even if the Directory Server instance is configured to run over TLS. This port number is used only for identification of the Directory Server instance in the Console; it does not specify the actual port number or protocol that is used for replication.

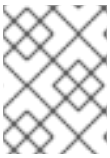
15.21. SETTING REPLICATION TIMEOUT PERIODS

Suppliers must have an exclusive connection to a consumer to send updates to the directory. As mentioned in Section 15.6.4, "Preventing Monopolization of a Consumer in Multi-Master Replication" , it is possible to configure a wait time for suppliers attempting to connect to a consumer, so that the supplier does not hang while the consumer is tied up with another supplier.

It is also possible to set a timeout period for a supplier, so that it does not stay connected to a consumer interminably attempting to send updates over a slow or broken connection.

There are two attributes which set the timeout period:

- ***nsDS5ReplicaTimeout*** sets the number of seconds that the replication operation waits for a response from the consumer before timing out and failing. To set the optimum number, check the access logs to see the average amount of time that the replication process takes, and set the timeout period accordingly.
- ***nsDS5DebugReplicaTimeout*** sets the timeout period for the replication operation when debug logging is enabled. This setting may be appreciably higher than the ***nsDS5ReplicaTimeout*** setting because debug logging can slow down directory operations. This attribute can optionally set an error log level where this parameter is applied; the default is replication debugging (8192).



NOTE

The timeout period is limited to the maximum 32-bit integer in seconds, which translates to 24.8 days.

Both of these attributes are set in the configuration for the replicated suffix. For example, this sets timeout periods for the **ou=People** suffix:

```
# ldapmodify -D "cn=Directory Manager" -W -x

dn: cn=replica,cn="ou=People,dc=example,dc=com",cn=mapping tree,cn=config
changetype: modify
add: nsDS5ReplicaTimeout
nsDS5ReplicaTimeout: 600
add: nsDS5DebugReplicaTimeout
nsDS5DebugReplicaTimeout: 6000
```

15.22. REPLICATING O=NETSCAPEROOT FOR ADMINISTRATION SERVER FAILOVER

Replication usually occurs between Directory Server user databases to distribute directory data, but it is also possible to use replication to provide failover support for the Administration Server database, **o=NetscapeRoot**.

1. Install and configure the first Directory Server instance.

   The **setup-ds-admin.pl** script has an option, **-f**, which references an **inf**. The **inf** can be used to import LDIF files through the **ConfigFile** parameter, and the LDIF files can create databases, suffixes, and replication entries. (The **inf** file is described in more detail in the *Red Hat Directory Server Installation Guide*.)

   ```
   # setup-ds-admin.pl -f /tmp/server1.inf
   ```

To configure the **o=NetscapeRoot** database on **server1** as a multi-master supplier replica, use the following statements in the **inf** file:

```
[slapd]
...
ConfigFile = repluser.ldif Example 15.4, “Example Supplier Bind DN Entry”
ConfigFile = changelog.ldif Example 15.1, “Example Changelog Entry”
ConfigFile = replica.ldif Example 15.2, “Example Supplier Replica Entry”
ConfigFile = replagreement.ldif Example 15.3, “Example Replication Agreement Entry”
...
```

2. Install and configure the second Directory Server instance. For the second server, **server2.example.com**, use the **setup-ds.pl** command, which installs a Directory Server instance without installing a local Administration Server.

```
# setup-ds.pl -f /tmp/server2.inf
```

With server2, use the **inf** file to create and configure a **o=NetscapeRoot** database on **server2** as a multi-master supplier replica:

```
[slapd]
...
ConfigFile = netscaperootdb.ldif Section 2.1.1.3, “Creating Root and Sub Suffixes using the Command Line”
ConfigFile = repluser.ldif Example 15.4, “Example Supplier Bind DN Entry”
ConfigFile = changelog.ldif Example 15.1, “Example Changelog Entry”
ConfigFile = replica.ldif Example 15.2, “Example Supplier Replica Entry”
ConfigFile = replagreement.ldif Example 15.3, “Example Replication Agreement Entry”
...
```

3. Initialize the **o=NetscapeRoot** database on **server2** from **server1**. Add the ***nsds5replicarefresh*** attribute to the replication agreement on **server1**.

```
# ldapmodify -D "cn=Directory Manager" -W -x -h supplier1.example.com

dn: cn=ExampleAgreement1,cn=replica,cn=o=NetscapeRoot,cn=mapping tree,cn=config
changetype: modify
replace: nsds5beginreplicarefresh
nsds5beginreplicarefresh: start
```

4. Run the **register-ds-admin.pl** to create a local Administration Server on **server2** and switch the configuration directory for **server2** to its own **o=NetscapeRoot** database from **server1**.

```
# register-ds-admin.pl
```

5. Add the following access control instructions (ACI) on **server2**, to enable members of the **Configuration Administrators Group**, the server instance entry **SIE group**, and the **admin** user, to run on suffixes belonging to **server2**. For example, to run on the **dc=example,dc=com** suffix, enter:

```
# ldapmodify -D "cn=Directory Manager" -W -x -h server2.example.com
dn: dc=example,dc=com
```

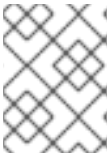
```
changetype: modify
add: aci
aci: (targetattr="*")(version 3.0; acl "Configuration Administrators Group";
 allow (all) groupdn="ldap:///cn=Configuration Administrators,ou=Groups,
 ou=TopologyManagement,o=NetscapeRoot";)
-
add: aci
aci: (targetattr="*")(version 3.0; acl "Configuration Administrator";
 allow (all) userdn="ldap:///uid=admin,
 ou=Administrators,ou=TopologyManagement,o=NetscapeRoot";)
-
add: aci
aci: (targetattr = "*")(version 3.0; acl "SIE Group";  allow (all) groupdn =
 "ldap:///cn=slapd-instance,cn=Red Hat Directory Server,cn=Server Group,
 cn=machine_name,ou=example.com,o=NetscapeRoot";)
```

6. Disable the PTA Plug-in on **server2** so that it does not pass bind operations for the administrative users in its **o=NetscapeRoot** to **server1**.

   See Section 1.9.2.2, "Enabling Plug-ins in the Directory Server Console" .

15.23. USING THE RETRO CHANGELOG PLUG-IN

The Retro Changelog plug-in configures Directory Server to maintain a changelog that is compatible with the changelog implemented in Directory Server 4.x.



NOTE

Only enable the Retro Changelog plug-in if you need to maintain a changelog for directory clients that depend on a Directory Server 4.x-style changelog.

To use the retro changelog plug-in, the Directory Server instance must be configured as a single-master replica.

When the Directory Server is configured to maintain a retro changelog, this changelog is stored in a separate database under a special suffix, **cn=changelog**.

The retro changelog consists of a single level of entries. Each entry in the changelog has the object class ***changeLogEntry***. For a list of possible attributes in a changelog entry, see the *Changelog Attributes* section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

15.23.1. Enabling the Retro Changelog Plug-in

The retro changelog plug-in configuration information is stored in the **cn=Retro Changelog Plugin,cn=plugins,cn=config** entry in **dse.ldif**. To enable the retro changelog plug-in from the command line:

1. Create an LDIF file that contains the following LDIF update statements:

```
dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

2. 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 -f retro.ldif
```

3. Restart the server.

   For information on restarting the server, see Section 1.4, “Starting and Stopping a Directory Server Instance”.

The retro changelog is created in the directory tree under a special suffix, **cn=changelog**.

The procedure for enabling the retro changelog plug-in from Directory Server Console is the same as for all Directory Server plug-ins. For information, see Section 1.9.2.2, “Enabling Plug-ins in the Directory Server Console”.

15.23.2. Trimming the Retro Changelog

The size of the retro changelog is automatically reduced if you lower the maximum age of records set in the ***nsslapd-changelogmaxage*** parameter and the next trim interval, set in ***nsslapd-changelog-trim-interval***, is executed.

For example, to set maximum age of records in the retro changelog to two days:

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

dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 2d
```

15.23.3. Searching and Modifying the Retro Changelog

The changelog supports search operations and is optimized for searches that include filters of the form **(&(changeNumber>=X)(changeNumber<=Y))**.

As a general rule, do not perform add or modify operations on the retro changelog entries, although entries can be deleted to trim the size of the changelog. Only modify the retro changelog entry to modify the default access control policy.

15.23.4. Retro Changelog and the Access Control Policy

When the retro changelog is created, the following access control rules apply by default:

- Read, search, and compare rights are granted to all authenticated users (**userdn=anyone**, not to be confused with anonymous access where **userdn=all**) to the retro changelog top entry **cn=changelog**.
- Write and delete access are not granted, except implicitly to the Directory Manager.

Do not grant read access to anonymous users because the changelog entries can contain modifications to sensitive information, such as passwords. Only authenticated applications and users should be allowed to access this information.

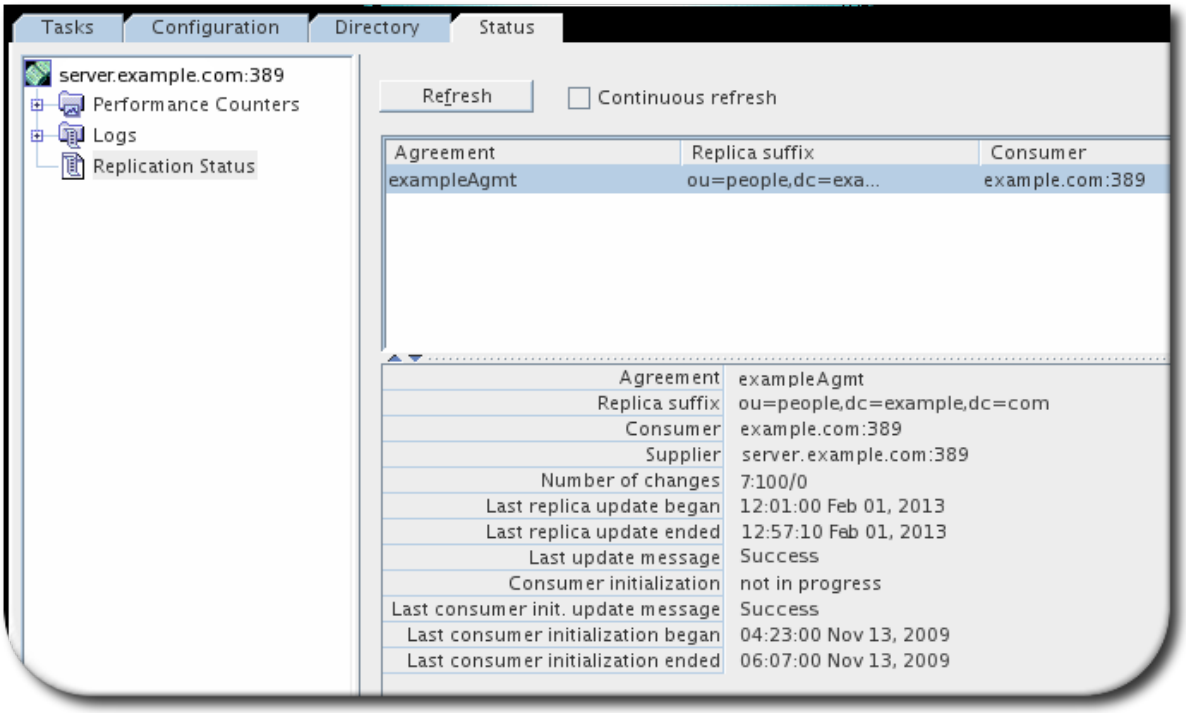
To modify the default access control policy which applies to the retro changelog, modify the ***aci*** attribute of the **cn=changelog** entry.

15.24. MONITORING REPLICATION STATUS

The replication status can be viewed in the Directory Server Console, Red Hat Administration Express (Section 15.24.2, "Monitoring Replication from Admin Express"), or from the command line.

15.24.1. Monitoring Replication Status from the Console

1. Select the **Status** tab, and then, in the left navigation tree, select **Replication Status**.



   In the right pane, a table appears that contains information about each of the replication agreements configured for this server.

2. Click **Refresh** to update the contents of the tab.

   The status information displayed is described in Table 15.1, "Directory Server Console Replication Status".

Table 15.1. Directory Server Console Replication Status

Table Header	Description
Agreement	The name of the replication agreement.
Replica suffix	The suffix that is replicated.
Supplier	The supplier server in the agreement.
Consumer	The consumer server in the agreement.

Table Header	Description
Number of changes	A ratio showing the changes sent to this replica since the server started. This value has the format *replica_id:changes_sent/changes_skipped*. So, if the replica ID is 7, 100 changes were sent, and no changes were skipped, the value of the number of changes is **7:100/0**.
Last replica update began	The time when the most recent replication update started.
Last replica update ended	The time when the most recent replication update ended.
Last update message	The status for the most recent replication updates.
Consumer initialization	The current status on consumer initialization (in progress or not).
Last consumer initialization update message	The status on the last initialization of the consumer.
Last consumer initialization began	The time when the initialization of the consumer replica started.
Last consumer initialization ended	The time when the initialization of the consumer replica ended.

15.24.2. Monitoring Replication from Admin Express

Admin Express has an option to monitor replication status in real-time, meaning that it shows the number of updates, times the most recent updates were sent, error and success messages, replication schedule, the replicated directory suffix, and other information. Unlike other ways of checking replication status, the Admin Express **Replication Status** page shows the real-time status of replication, including updates in progress, current changes sequence numbers, and the lag between when a change is made on the supplier and when that change is sent to the consumer.

Monitoring replication is set up using a simple configuration file which specifies which server to monitor and what supplier and consumer replicas to include in the status page.

When trying to monitor replication status through Admin Express, remember two things:

- The **Replication Status** page is only available for supplier servers. (It can be opened for other types of replicas; there is just no information available and has the message *The server is not a master or it has no replication agreement*.)
- The configuration file must be in a directory that is accessible to Administration Server, and the file must be readable by the Administration Server user. By default, the user is **dirsrv**.

  The user is set in the **console.conf** file. To check the user, use **grep** to return the value:

  ```
  # grep \^User /etc/dirsrv/admin-serv/console.conf
  ```

The configuration file should be readable by the Administration Server user and no other users, so consider resetting the permissions on the file:

```
# chmod 0400 filename
```

To view in-progress status of replication in Admin Express:

1. Create a configuration file. The configuration file lists all of the servers to monitor for replication, giving their host name or IPv4 or IPv6 address, port, the bind credentials to use, and then optional settings for aliases and time lag colors.

```
#Configuration File for Monitoring Replication Using Admin Express
 [connection] Required. Gives the server host (or IPv4 or IPv6 address), port, supplier bind
DN, and password.
host1.example.com:389:cn=replication manager:mypassword
host2.example.com:3891:cn=replication manager:altpassword

[alias] Optional. Gives a friendly-name alias to the servers and consumers.
M1 = host1.example.com:389
M2 = host2.example.com:3891
C1 = host3.example.com:3892
C2 = host4.example.com:3890

[color] Optional. Sets the color for the time lag boxes.
0 = #ccffcc
5 = #FFFFCC
60 = #FFCCCC
```

   The configuration file must be in a directory that is accessible to the Administration Server, and the file must be readable by the Administration Server user. By default, the user is **dirsrv**.

   The user is set in the **console.conf** file. To check the user, use **grep** to return the value:

```
# grep \^User /etc/dirsrv/admin-serv/console.conf
```

   The configuration file should be readable by the Administration Server user and no other users, so consider resetting the permissions on the file:

```
# chmod 0400 filename
```

2. In the Administration Server web page, click the **Admin Express** link, and log in.

3. Click the **Replication Status** link by the supplier server name.

4. Type the path to the configuration file in the **Configuration file** field. Also, set the refresh rate, which is how frequently the replication status page updates; the default is 300 seconds.

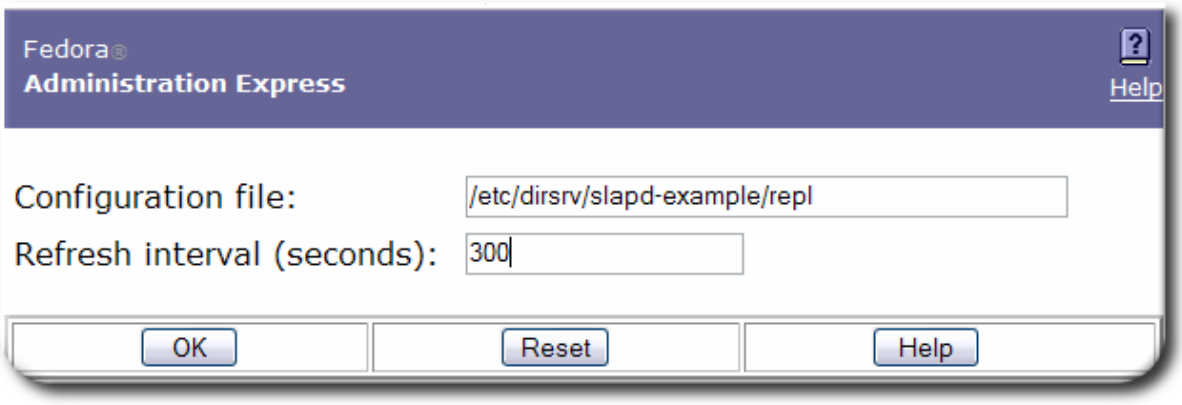


Figure 15.5. Viewing Replication Status

5. Click **OK**.

The **Replication Status** page shows the status for sending updates to every consumer listed in the configuration file.

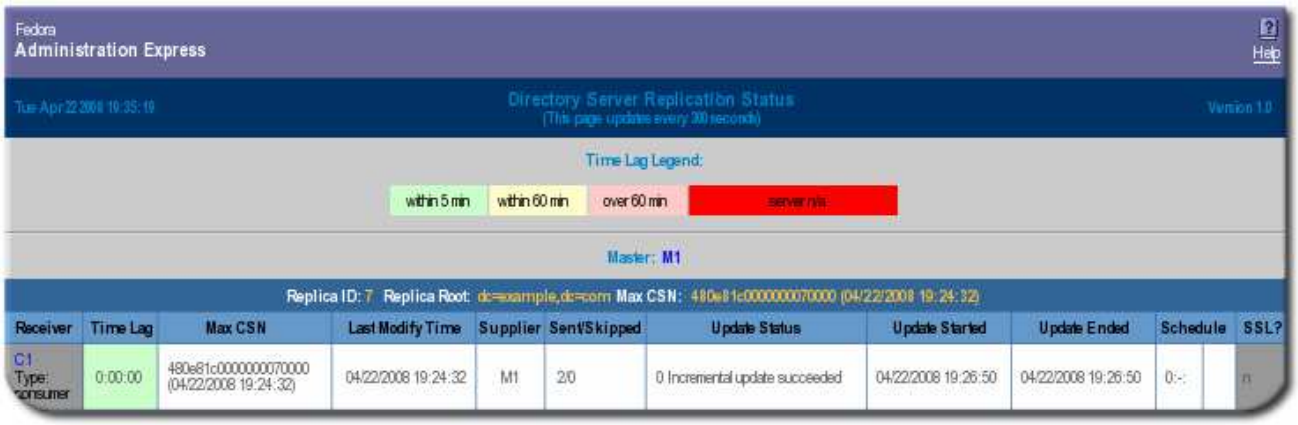


Figure 15.6. Viewing Replication Status

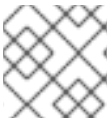
Table	Description
Table header	The table header shows the replica ID of the supplier replica, the replicated suffix root (such as **dc=example,dc=com**), and the maximum change state number (CSN) on the supplier. (The CSN is the ID of the latest change on the supplier, while the max CSN for the supplier shows the last update it received.)
Max CSN	The ID number of the most recent CSN the consumer has received that originated from the supplier.
Time lag	How long it takes for the consumer to receive updates from the supplier; this is the time difference between the supplier and the consumer's max CSNs. When a consumer is in sync with its supplier, the time lag is **0**.
Last Modify Time	Gives the time of the last update for the consumer (the time the last CSN entry was sent).

Table	Description
Supplier	Gives the name of the supplier sending updates to that consumer; this can be useful if a consumer receives updates from multiple suppliers or there are multiple suppliers being monitored on the **Replication Status** page.
Sent/Skipped	The number of changes that were sent from the supplier and the number skipped in the replication update. The numbers are kept in suppliers' memory only and are cleared if the supplier is restarted.
Update Status	The status code (and meaning) for the last update. This column can indicate a possible deadlock if *all* the suppliers complain that they cannot acquire a busy replica. It is normal for there to be a busy message if one of the suppliers is doing an update.
Update Start and End	The timestamps for when the most recent update process started and ended.
Schedule	The configured replication schedule. **0:-:** means that the consumer is continually updated by the supplier.
SSL?	Indicates whether the supplier connects to the consumer over TLS.

15.24.3. Monitoring Replication from the Command-Line

To display replication status from the command line, run the **/usr/bin/repl-monitor.pl** script with the **-s** option added. The script prints the report in plain text format and is useful, for example, in situations when the user wants to quickly determine the replication status but a browser is not available. Similarly to Admin Express described in Section 15.24.2, "Monitoring Replication from Admin Express" , **repl-monitor.pl** shows replication status in real-time.

The **repl-monitor.pl** script accepts a number of command-line options. For more information on how to use it, see the repl-monitor(1) man page or the Directory Server Configuration, Command, and File Reference guide.



NOTE

When run without the **-s** option, **repl-monitor.pl** generates the report as an HTML file.

15.25. COMPARING TWO DIRECTORY SERVER INSTANCES

In certain situations, an administrator wants to compare if two Directory Servers are synchronized. The **ds-replcheck** utility enables you to compare two online servers. Alternatively, **ds-replcheck** can compare two LDIF-formatted files in offline mode.

NOTE

To compare two databases offline, export them using the **db2ldif -r** command to include replication state information.

If you compare two online servers, the contents of the databases usually differ, if they are under heavy load. To work around this problem, the script uses a lag time value in by passing the ***-l time_in_seconds*** parameter to **ds-replcheck**. By default, this value is set to **300** seconds (5 minutes). If the utility detects an inconsistency that is within the lag time, it is not reported. This helps to reduce false positives.

By default, if you excluded certain attributes in the replication agreement from being replicated, **ds-replcheck** reports these attributes as different. To ignore these attributes, pass the ***-i attribute_list*** parameter to the utility.

For example, to compare the ***dc=example,dc=com*** suffix of two Directory Servers:

```
# ds-replcheck -D "cn=Directory Manager" -W \
    -m ldap://server1.example.com:389 \
    -r ldap://server2.example.com:389 \
    -b "dc=example,dc=com"
```

The output of the utility contains the following sections:

Database RUV's

Lists the Replication Update Vectors (RUV) of the databases including the minimum and maximum Change Sequence Numbers (CSN). For example:

```
Master RUV:
  {replica 1 ldap://server1.example.com:389} 58e53b92000200010000 58e6ab46000000010000
  {replica 2 ldap://server2.example.com:389} 58e53baa000000020000 58e69d7e000000020000
  {replicageneration} 58e53b7a000000010000

Replica RUV:
  {replica 1 ldap://server1.example.com:389} 58e53ba1000000010000 58e6ab46000000010000
  {replica 2 ldap://server2.example.com:389} 58e53baa000000020000 58e7e8a3000000020000
  {replicageneration} 58e53b7a000000010000
```

Entry Count

Displays the total number of entries on the both servers, including tombstone entries. For example:

```
Master:  12
Replica: 10
```

Tombstones

Displays the number of tombstone entries on each replica. These entries are added to the total entry count. For example:

```
Master:  4
Replica: 2
```

Conflict Entries

Lists the Distinguished Names (DN) of each conflict entry, the conflict type, and the date it was created. For example:

```
Master Conflict Entries: 1

 - nsuniqueid=48177227-2ab611e7-afcb801a-ecef6d49+uid=user1,dc=example,dc=com
```

```
   - Conflict:   namingConflict (add) uid=user1,dc=example,dc=com
   - Glue entry: no
   - Created:    Wed Apr 26 20:27:40 2017


Replica Conflict Entries: 1

 - nsuniqueid=48177227-2ab611e7-afcb801a-ecef6d49+uid=user1,dc=example,dc=com
   - Conflict:   namingConflict (add) uid=user1,dc=example,dc=com
   - Glue entry: no
   - Created:    Wed Apr 26 20:27:40 2017
```

Missing Entries

Lists the DNs of each missing entry and the creation date from the other server where the entry resides. For example:

```
Entries missing on Master:
 - uid=user2,dc=example,dc=com (Created on Replica at: Wed Apr 12 14:43:24 2017)
 - uid=user3,dc=example,dc=com (Created on Replica at: Wed Apr 12 14:43:24 2017)

Entries missing on Replica:
 - uid=user4,dc=example,dc=com (Created on Master at: Wed Apr 12 14:43:24 2017)
```

Entry Inconsistencies

Lists the DNs of the entry that contain attributes that are different to those on the other server. If a state information is available, it is also displayed. If no state information for an attribute is available, it is listed as an origin value. This means that the value was not updated since the replication was initialized for the first time. For example:

```
cn=group1,dc=example,dc=com
---------------------------
Replica missing attribute "objectclass":

 - Master's State Info: objectClass;vucsn-58e53baa000000020000: top
 - Date: Wed Apr  5 14:47:06 2017

 - Master's State Info: objectClass;vucsn-58e53baa000000020000: groupofuniquenames
 - Date: Wed Apr  5 14:47:06 2017
```

For further details about the **ds-replcheck** utility, see the description in the Red Hat Configuration, Command, and File Reference.

15.26. SOLVING COMMON REPLICATION CONFLICTS

Multi-master replication uses an eventually-consistency replication model. This means that the same entries can be changed on different servers. When replication occurs between these two servers, the conflicting changes need to be resolved. Mostly, resolution occurs automatically, based on the time stamp associated with the change on each server. The most recent change takes precedence.

However, there are some cases where conflicts require manual intervention in order to reach a resolution. Entries with a change conflict that cannot be resolved automatically by the replication process contain the ***nsds5ReplConflict*** conflict marker attribute and the ***ldapSubEntry*** object class.

The ***nsds5ReplConflict*** attribute is an operational attribute which is indexed for presence and equality.

To list conflict entries:

```
# ldapsearch -D "cn=Directory Manager" -W -b "dc=example,dc=com" \
    "(&(objectClass=ldapSubEntry)(nsds5ReplConflict=*))" \* nsds5ReplConflict
```

15.26.1. Solving Naming Conflicts

When two entries are created with the same DN on different servers, the automatic conflict resolution procedure during replication renames the last entry created, including the entry's unique identifier in the DN. Every directory entry includes a unique identifier stored in the ***nsuniqueid*** operational attribute. When a naming conflict occurs, this unique ID is appended to the non-unique DN.

For example, if the **uid=*user_name*,ou=People,dc=example,dc=com** entry was created on two different servers, replication adds the unique ID to the DN of the last entry created. This means, the following entries exist:

- **uid=*user_name*,dc=example,dc=com**
- **nsuniqueid=*66446001-1dd211b2*+uid=*user_name*,dc=example,dc=com**

To resolve the replication conflict, you must manually decide how to proceed:

- To keep only the valid entry (**uid=*user_name*,dc=example,dc=com**), by deleting the conflict entry:

  ```
  # ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
      uid=nsuniqueid=66446001-1dd211b2+user_name,dc=example,dc=com
  ```

- Keep only the conflict entry (**nsuniqueid=*66446001-1dd211b2*+uid=*user_name*,dc=example,dc=com**):

  1. Delete the valid entry:

     ```
     # ldapdelete -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
         uid=user_name,dc=example,dc=com
     ```

  2. Rename the conflict entry. See Section 15.26.1.1, “Renaming an Entry with a Multi-Valued Naming Attribute”.

- To keep both entries, rename the conflict entry. See Section 15.26.1.1, “Renaming an Entry with a Multi-Valued Naming Attribute”.

15.26.1.1. Renaming an Entry with a Multi-Valued Naming Attribute

To rename an entry that has a multi-valued naming attribute:

1. Rename the entry using a new value for the naming attribute, and keep the old RDN. For example:

   ```
   # ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
    dn: nsuniqueid=66446001-1dd211b2+uid=adamss,dc=example,dc=com
    changetype: modrdn
   ```

```
newrdn: uid=NewValue
deleteoldrdn: 0
```

For further details about keeping an RDN when renaming an entry, see Section 3.1.6.3, "The ***deleteOldRDN*** Parameter".

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
 dn: uid=NewValue,dc=example,dc=com
 changetype: modify
 delete: uid
 uid: adamss
 -
 delete: nsds5ReplConflict
 -
```

NOTE

The unique identifier attribute ***nsuniqueid*** cannot be deleted.

The Console does not support editing multi-valued RDNs. For example, if there are two servers in a multi-master mode, an entry can be created on each server with the same user ID, and then the new entries' RDN changed to the ***nsuniqueid uid*** value. Attempting to modify this entry from the Console returns the error *Changes cannot be saved for entries with multi-valued RDNs* .

Opening the entry in the advanced mode shows that the naming attribute has been set to ***nsuniqueid uid***. However, the entry cannot be changed or corrected by changing the user ID and RDN values to something different. For example, if **jdoe** was the user ID and it should be changed to **jdoe1**, it cannot be done from the Console. Instead, use the **ldapmodify** command:

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

dn: cn=John Doe
changetype: modify
replace: uid
uid: jdoe

dn: cn=John Doe
changetype: modrdn
newrdn: uid=jdoe1
deleteoldrdn: 1
```

15.26.1.2. Renaming an Entry with a Single-Valued Naming Attribute

To rename an entry that has a single-valued naming attribute:

1. Rename the entry using a different naming attribute, and keep the old RDN. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
 dn: nsuniqueid=66446001-1dd211b2+dc=pubs,dc=example,dc=com
 changetype: modrdn
```

```
newrdn: cn=TempValue
deleteoldrdn: 0
```

For further details about keeping an RDN when renaming an entry, see Section 3.1.6.3, "The ***deleteOldRDN*** Parameter".

2. Remove the old RDN value of the naming attribute and the conflict marker attribute. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
 dn: cn=TempValue,dc=example,dc=com
 changetype: modify
 delete: dc
 dc: pubs
 -
 delete: nsds5ReplConflict
 -
```

NOTE

The unique identifier attribute ***nsuniqueid*** cannot be deleted.

3. Rename the entry with the intended attribute-value pair. For example:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
 dn: cn=TempValue,dc=example,dc=com
 changetype: modrdn
 newrdn: dc=NewValue
 deleteoldrdn: 1
```

Setting the value of the ***deleteoldrdn*** attribute to **1** deletes the temporary attribute-value pair **cn=***TempValue*. To keep this attribute, set the value of the ***deleteoldrdn*** attribute to **0**.

15.26.2. Solving Orphan Entry Conflicts

When a delete operation is replicated and the consumer server finds that the entry to be deleted has child entries, the conflict resolution procedure creates a **glue** entry to avoid having orphaned entries in the directory.

In the same way, when an add operation is replicated and the consumer server cannot find the parent entry, the conflict resolution procedure creates a glue entry representing the parent so that the new entry is not an orphan entry.

Glue entries are temporary entries that include the object classes **glue** and **extensibleObject**. Glue entries can be created in several ways:

- If the conflict resolution procedure finds a deleted entry with a matching unique identifier, the glue entry is a resurrection of that entry, with the addition of the **glue** object class and the ***nsds5ReplConflict*** attribute.

  In such cases, either modify the glue entry to remove the **glue** object class and the ***nsds5ReplConflict*** attribute to keep the entry as a normal entry or delete the glue entry and its child entries.

- The server creates a minimalistic entry with the **glue** and **extensibleObject** object classes.

In such cases, modify the entry to turn it into a meaningful entry or delete it and all of its child entries.

15.26.3. Resolving Errors for Obsolete or Missing Suppliers

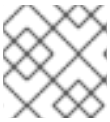
Information about the replication topology, that is all suppliers which supply updates to each other and other replicas within the same replication group, is contained in a set of metadata called the *replica update vector (RUV)*. The RUV contains information about the supplier such as its ID and URL, its latest change state number (CSN) on the local server, and the CSN of the first change. Both suppliers and consumers store RUV information, and they use it to control replication updates.

When one supplier is removed from the replication topology, it may remain in another replica's RUV. When the other replica is restarted, it can record errors in its log, warning that the replication plug-in does not recognize the removed supplier. The errors will look similar to the following example:

```
[22/Jan/2020:17:16:01 -0500] NSMMReplicationPlugin - ruv_compare_ruv: RUV [changelog max
RUV] does not contain element [{replica 8 ldap://m2.example.com:389} 4aac3e59000000080000
4c6f2a02000000080000] which is present in RUV [database RUV]

<...several more samples...>

[22/Jan/2020:17:16:01 -0500] NSMMReplicationPlugin - replica_check_for_data_reload: Warning: for
replica dc=example,dc=com there were some differences between the changelog max RUV and the
database RUV. If there are obsolete elements in the database RUV, you should remove them using
the CLEANALLRUV task. If they are not obsolete, you should check their status to see why there are
no changes from those servers in the changelog.
```

Note which replica and its ID; in this case, replica **8**.

When the supplier is permanently removed from the topology, then any lingering metadata about that supplier should be purged from every other supplier's RUV entry. Use the **cleanallruv** directory tasks to remove a RUV entry from all suppliers in the topology.



NOTE

The **cleanallruv** task is replicated. Therefore, you only need to run it on one master.

Procedure 15.1. Removing an Obsolete or Missing Supplier Using the cleanallruv Task Operation

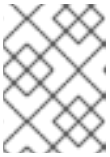
1. List all RUV records and replica IDs, both valid and invalid, as deleted masters may have left metadata on other masters:

```
# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com -D "cn=Directory Manager" -W -b
dc=example,dc=com '(&(nsuniqueid=ffffffff-ffffffff-ffffffff-ffffffff)(objectclass=nstombstone))'
nsDS5ReplicaId nsDS5ReplicaType nsds50ruv
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5ReplicaId: 1
nsDS5ReplicaType: 3
nsds50ruv: {replicageneration} 55d5093a000000010000
nsds50ruv: {replica 1 ldap://m1.example.com:389} 55d57026000000010000
55d57275000000010000
nsds50ruv: {replica 20 ldap://m2.example.com:389} 55e74b8c000000140000
55e74bf7000000140000
```

```
nsds50ruv: {replica 9 ldap://m2.example.com:389}
nsds50ruv: {replica 8 ldap://m2.example.com:389} 506f921f000000080000
50774211000500080000
```

Note the returned replica IDs: **1**, **20**, **9**, and **8**.

2. List the currently defined and valid replica IDs of all masters which are replicating databases by searching the replica configuration entries DN **cn=replica** under the **cn=config** suffix.



NOTE

Consumers and read-only nodes always have the replica ID set to **65535**, and **nsDS5ReplicaType: 3** signifies a master.

```
# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com m2.example.com -D "cn=Directory
Manager" -W -b cn=config cn=replica nsDS5ReplicaId nsDS5ReplicaType
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5ReplicaId: 1
nsDS5ReplicaType: 3

dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5ReplicaId: 20
nsDS5ReplicaType: 3
```

After you search all URIs returned in the first step (in this procedure, **m1.example.com** and **m2.example.com**), compare the list of returned masters (entries which have **nsDS5ReplicaType: 3**) to the list of RUVs from the previous step. In the above example, this search only returned IDs **1** and **20**, but the previous search also returned **9** and **8** on URI **m2.example.com**. This means that the latter two are removed, and their RUVs need to be cleaned.

3. After determining which RUVs require cleaning, create a new **cn=cleanallruv,cn=tasks,cn=config** entry and provide the following information about your replication configuration:

   - The base DN of the replicated database (**replica-base-dn**)
   - The replica ID (**replica-id**)
   - Whether to catch up to the maximum change state number (CSN) from the missing supplier, or whether to just remove all RUV entries and miss any updates (**replica-force-cleaning**); setting this attribute to **no** means that the task will wait for all the configured replicas to catch up with all the changes from the removed replica first, and then remove the RUV.

```
# ldapmodify -a -D "cn=Directory Manager" -W -H m2.example.com -x
dn: cn=clean 8,cn=cleanallruv,cn=tasks,cn=config
objectclass: extensibleObject
replica-base-dn: dc=example,dc=com
replica-id: 8
replica-force-cleaning: no
cn: clean 8
```

NOTE

The **cleanallruv** task is replicated. Therefore, you only need to run it on one master.

Repeat the same for every RUV you want to clean (ID **9** in this procedure).

4. After cleaning the RUVs of all replicas discovered earlier, you can again use the search from the first step to verify that all extra RUVs are removed:

```
# ldapsearch -o ldif-wrap=no -xLLL -H m1.example.com -D "cn=Directory Manager" -W -b
dc=example,dc=com '(&(nsuniqueid=ffffffff-ffffffff-ffffffff-ffffffff)(objectclass=nstombstone))'
nsDS5ReplicaId nsDS5ReplicaType nsds50ruv
dn: cn=replica,cn=dc\3Dexample\2Cdc\3Dcom,cn=mapping tree,cn=config
nsDS5ReplicaId: 1
nsDS5ReplicaType: 3
nsds50ruv: {replicageneration} 55d5093a000000010000
nsds50ruv: {replica 1 ldap://m1.example.com:389} 55d57026000000010000
55d57275000000010000
nsds50ruv: {replica 20 ldap://m2.example.com:389} 55e74b8c000000140000
55e74bf7000000140000
```

As you can see in the above output, replica IDs **8** and **9** are no longer present, which indicates that their RUVs have been cleaned successfully.

15.27. TROUBLESHOOTING REPLICATION-RELATED PROBLEMS

This section lists some error messages, explains possible causes, and offers remedies.

It is possible to get more debugging information for replication by setting the error log level to **8192**, which is replication debugging. See Section 20.3.7, “Configuring Log Levels”.

To change the error log level to **8192** with **ldapmodify**:

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

dn: cn=config
changetype: modify
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 8192
```

Because log level is additive, running the above command will result in excessive messages in the error log. So, use it judiciously.

To turn off replication debugging log, set the same attribute to **0**.

The **cl-dump.pl** script, which is explained in detail in the *Red Hat Directory Server Configuration, Command, and File Reference* can also help troubleshoot replication-related problems. Depending on the usage options, the script can selectively dump a particular replica:

- Dump the contents of a **replication-change-log** file and in-memory variables **purge RUV** and **maxRUV**.
- Grep and interpret change state numbers (CSNs) in the changelog.

- Get the base-64 encoded changelog from the Directory Server, and then decode the changelog.

The following sections describe many common replication problems.

agmt=%s (%s:%d) Replica has a different generation ID than the local data

- Reason: The consumer specified at the beginning of this message has not been (successfully) initialized yet, or it was initialized from a different root supplier.
- Impact: The local supplier will not replicate any data to the consumer.
- Remedy: Ignore this message if it occurs before the consumer is initialized. Otherwise, reinitialize the consumer if the message is persistent. In a multi-master environment, all the servers should be initialized only once from a root supplier, directly or indirectly. For example, M1 initializes M2 and M4, M2 then initializes M3, and so on. The important thing to note is that M2 must not start initializing M3 until M2's own initialization is done (check the total update status from the M1's Console or M1 or M2's error log). Also, M2 should not initialize M1 back.

Warning: data for replica's was reloaded, and it no longer matches the data in the changelog. Recreating the changelog file. This could affect replication with replica's consumers, in which case the consumers should be reinitialized.

- Reason: This message may appear only when a supplier is restarted. It indicates that the supplier was unable to write the changelog or did not flush out its RUV at its last shutdown. The former is usually because of a disk-space problem, and the latter because a server crashed or was ungracefully shut down.
- Impact: The server will not be able to send the changes to a consumer if the consumer's ***maxcsn*** no longer exists in the server's changelog.
- Remedy: Check the disk space and the possible core file (under the server's logs directory). If this is a single-master replication, reinitialize the consumers. Otherwise, if the server later complains that it cannot locate some CSN for a consumer, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

agmt=%s(%s:%d): Can't locate CSN %s in the changelog (DB rc=%d). The consumer may need to be reinitialized.

- Reason: Most likely the changelog was recreated because of the disk is full or the server ungracefully shutdown.
- Impact: The local server will not be able to send any more change to that consumer until the consumer is reinitialized or gets the CSN from other suppliers.
- Remedy: If this is a single-master replication, reinitialize the consumers. Otherwise, see if the consumer can get the CSN from other suppliers. If not, reinitialize the consumer.

Too much time skew

- Reason: The system clocks on the host machines are extremely out of sync.
- Impact: The system clock is used to generate a part of the CSN. In order to reflect the change sequence among multiple suppliers, suppliers would forward-adjust their local clocks based on the remote clocks of the other suppliers. Because the adjustment is limited to a certain amount, any difference that exceeds the permitted limit will cause the replication session to be aborted.

- Remedy: Synchronize the system clocks on the Directory Server host machines. If applicable, run the network time protocol (**ntp**) daemon on those hosts.

agmt=%s(%s:%d): Warning: Unable to send endReplication extended operation (%s)

- Reason: The consumer is not responding.

- Impact: If the consumer recovers without being restarted, there is a chance that the replica on the consumer will be locked forever if it did not receive the release lock message from the supplier.

- Remedy: Watch if the consumer can receive any new change from any of its suppliers, or start the replication monitor, and see if all the suppliers of this consumer warn that the replica is busy. If the replica appears to be locked forever and no supplier can get in, restart the consumer.

Changelog is getting too big.

- Reason: Either changelog purge is turned off, which is the default setting, or changelog purge is turned on, but some consumers are way behind the supplier.

- Remedy: By default changelog purge is turned off. To turn it on from the command line, run **ldapmodify** as follows:

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

dn: cn=changelog5,cn=config
changetype: modify
add: nsslapd-changelogmaxage
nsslapd-changelogmaxage: 1d
```

1d means 1 day. Other valid time units are s for seconds, m for minutes, h for hours, and w for weeks. A value of 0 turns off the purge.

With changelog purge turned on, a purge thread that wakes up every five minutes will remove a change if its age is greater than the value of nsslapd-changelogmaxage and if it has been replayed to all the direct consumers of this supplier (supplier or hub).

If it appears that the changelog is not purged when the purge threshold is reached, check the maximum time lag from the replication monitor among all the consumers. Irrespective of what the purge threshold is, no change will be purged before it is replayed by all the consumers.

The Replication Monitor is not responding.

- Reason: The LDAPS port is specified in some replication agreement, but the certificate database is not specified or not accessible by the Replication Monitor. If there is no LDAPS port problem, one of the servers in the replication topology might hang.

- Remedy: Map the TLS port to a non-TLS port in the configuration file of the Replication Monitor. For example, if 636 is the TLS port and 389 is the non-TLS port, add the following line in the **[connection]** section:

```
*:636=389:*:password
```

In the Replication Monitor, some consumers show just the header of the table.

- Reason: No change has originated from the corresponding suppliers. In this case, the ***MaxCSN***: in the header part should be **"None"**

in the header part should be "**None**".

- Remedy: There is nothing wrong if there is no change originated from a supplier.

CHAPTER 16. SYNCHRONIZING RED HAT DIRECTORY SERVER WITH MICROSOFT ACTIVE DIRECTORY

Windows Synchronization 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.

16.1. ABOUT WINDOWS SYNCHRONIZATION

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 Synchronization has two parts, one for user and group entries and the other for passwords:

- *Directory Server Windows Synchronization.* 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 synchronized (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 Synchronization Service.* Password changes made on Directory Server are automatically synchronized 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 Synchronization Service. This application captures password changes on the Windows machines and send them to the Directory Server over LDAPS.

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

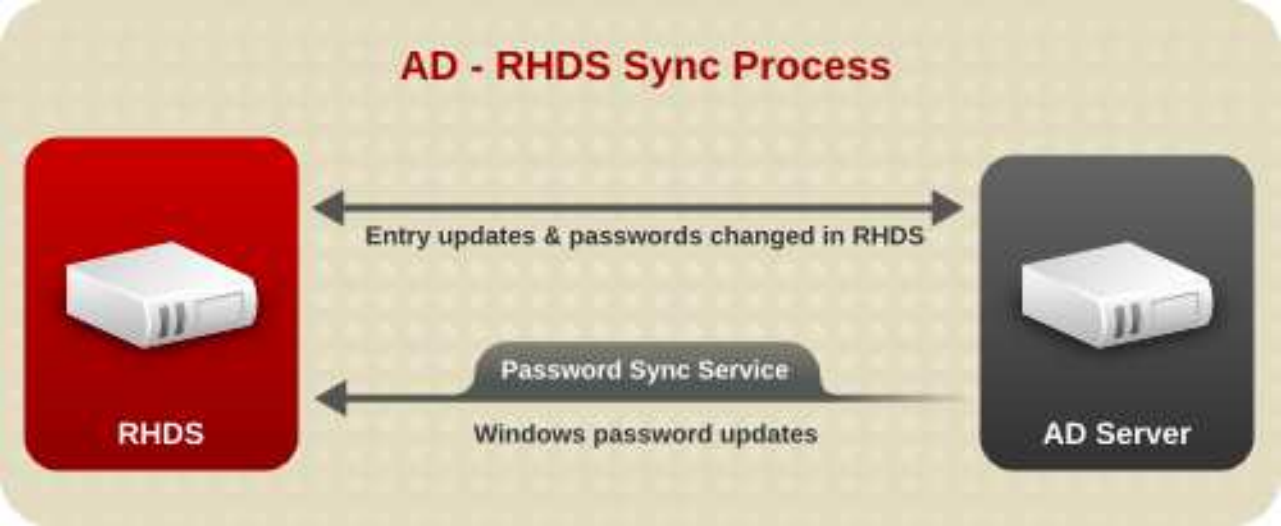


Figure 16.1. Active Directory – Directory Server Synchronization Process

Synchronization is configured and controlled by one or more *synchronization agreements*, which establishes synchronization between *sync peers*, the directory servers being synchronized. 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 Synchronization Service will not send passwords from the Active Directory domain to the Directory Server. Windows Synchronization allows both LDAPS using TLS and Start TLS.

Multiple subtree pairs can be configured to sync each other. Unlike replication, which connects *databases*, synchronization is between *suffixes*, parts of the directory tree structure. The synchronized 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 Synchronization 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 Synchronization 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. Directory Server runs the Dirsync search periodically by default every five minutes to check for changes on the Active Directory server. You can change this default by setting the ***winSyncInterval*** parameter in the **cn=*syncAgreement_Name*,cn=WindowsReplica,cn=*suffix_Name*,cn=mapping tree,cn=config** entry. 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 16.11, “Sending Synchronization Updates” for more information.

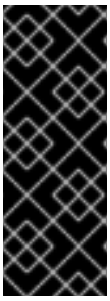
Windows Synchronization provides some control over which entries are synchronized to grant administrators fine-grained control of the entries that are synchronized and to give sufficient flexibility to support different deployment scenarios. This control is set through different configuration attributes 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 16.2, “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.

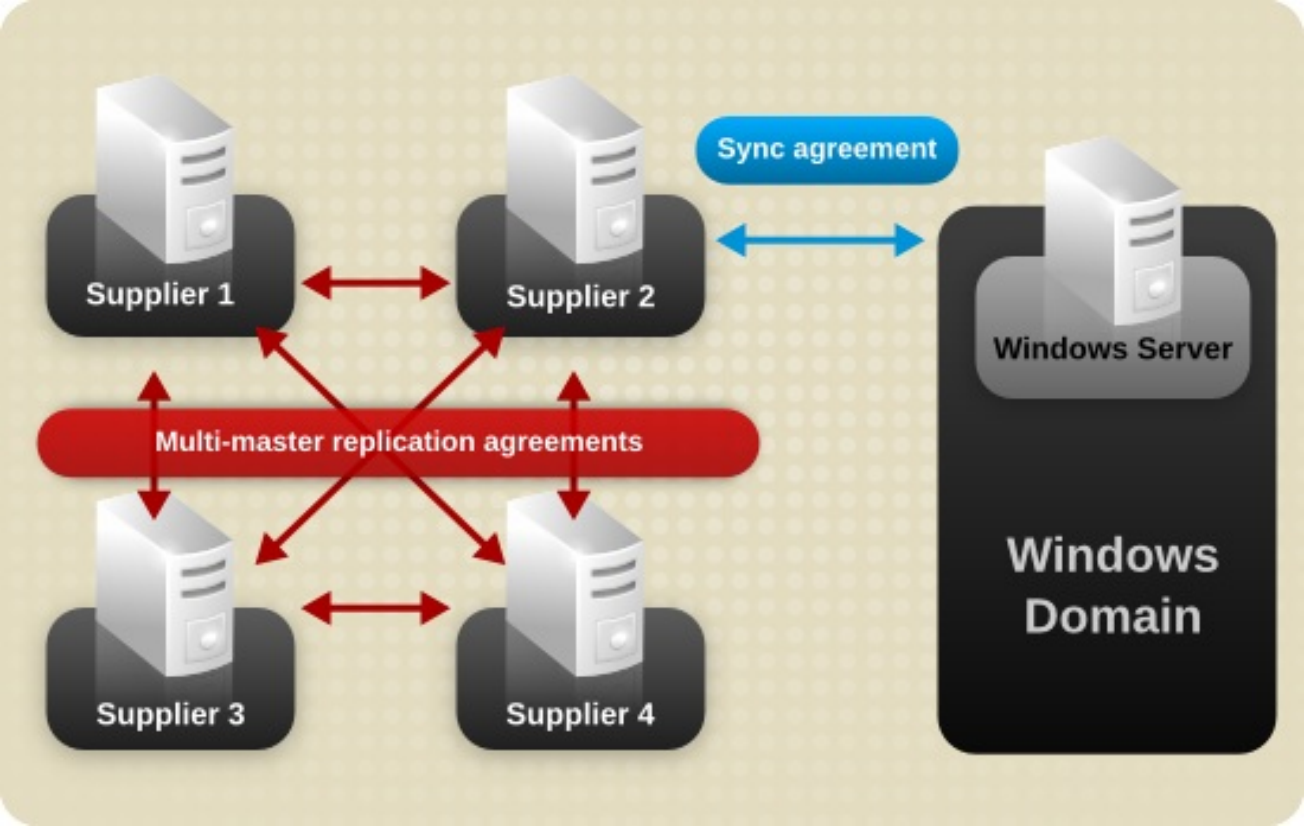


Figure 16.2. Multi-Master Directory Server – Windows Domain Synchronization

Directory Server passwords are synchronized along with other entry attributes because plain-text passwords are retained in the Directory Server changelog. The Password Synchronization service is needed to catch password changes made on Active Directory. Without the Password Synchronization 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.

16.2. SUPPORTED ACTIVE DIRECTORY VERSIONS

See *Red Hat Directory Server Release Notes*.

16.3. 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 synchronized 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 16, *Synchronizing Red Hat Directory Server with Microsoft Active Directory* for more information on synchronizing Directory Server and Windows users and passwords.

16.4. STEPS FOR CONFIGURING WINDOWS SYNCHRONIZATION

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 TLS. Therefore, this configuration section assumes that TLS must also be configured.

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

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

16.4.1. Step 1: Configure TLS on Directory Server

The full instructions for configuring the Directory Server to run in TLS are at Section 9.4.1, "Enabling TLS in Directory Server". Basically, the Directory Server needs to have the appropriate TLS certificates installed, be configured to run over an LDAPS port, and allow client authentication from other servers.

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

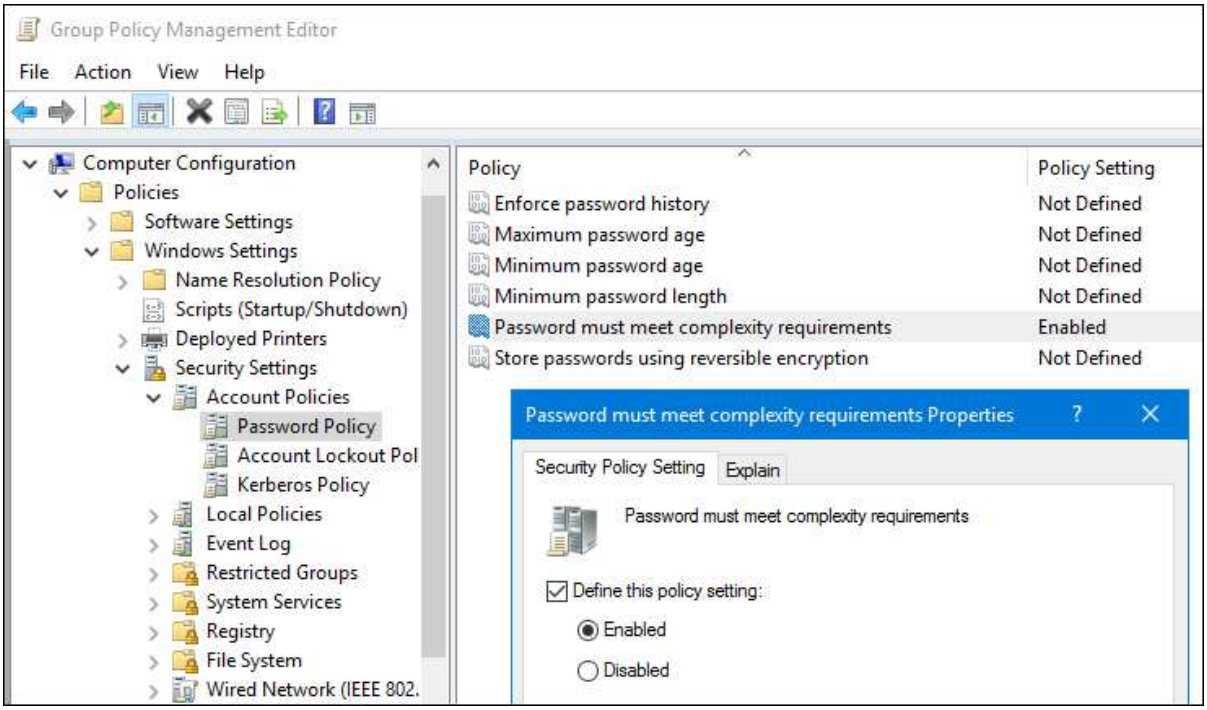
- CA certificate, shared between the Directory Server and AD
- Server certificates for the Directory Server and AD sync peers, which are accessible by the sync services

16.4.2. Step 2: Configure the Active Directory Domain

You can configure synchronization only with AD domain controllers. Additionally, password complexity must be enabled in AD.

To enable password complexity:

1. Open the **Group Policy Management** console and create a new Group Policy Object (GPO). For details, see the Windows documentation.
2. Right-click the GPO, and select **Edit** to open the **Group Policy Management Editor**.
3. Navigate to **Computer Configuration → Windows Settings → Security Settings → Account Policies → Password Policy**, and double-click the policy named **Password must meet complexity requirements**.
4. Enable the policy and click **OK**.



5. Close the **Group Policy Management Editor** and the **Group Policy Management** console.

Configure TLS and set up a root CA on the AD 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 AD server.

2. Set up the AD server to use the TLS server certificate.

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

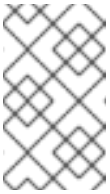
```
;----------------- request.inf -----------------

[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

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

    2. Generate the certificate request.

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

    3. Submit the request to the AD CA. For example:

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



NOTE

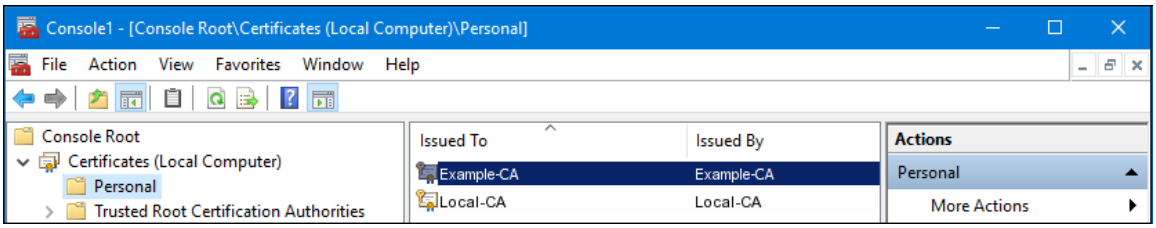
If the command-line tool returns an error message, then use the Web browser to access the CA and submit the certificate request. If IIS is running, then the CA URL is **http://***servername***/certsrv**.

    4. Accept the certificate request. For example:

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

3. Make sure that the server certificate is present on the AD 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**.



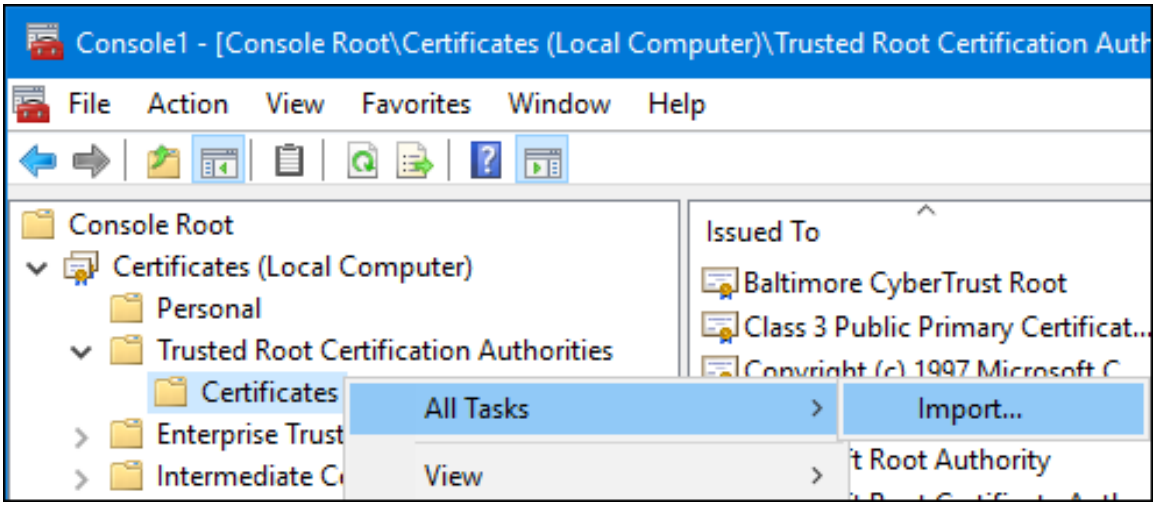
5. The new certificate should be listed with the other certificates.

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

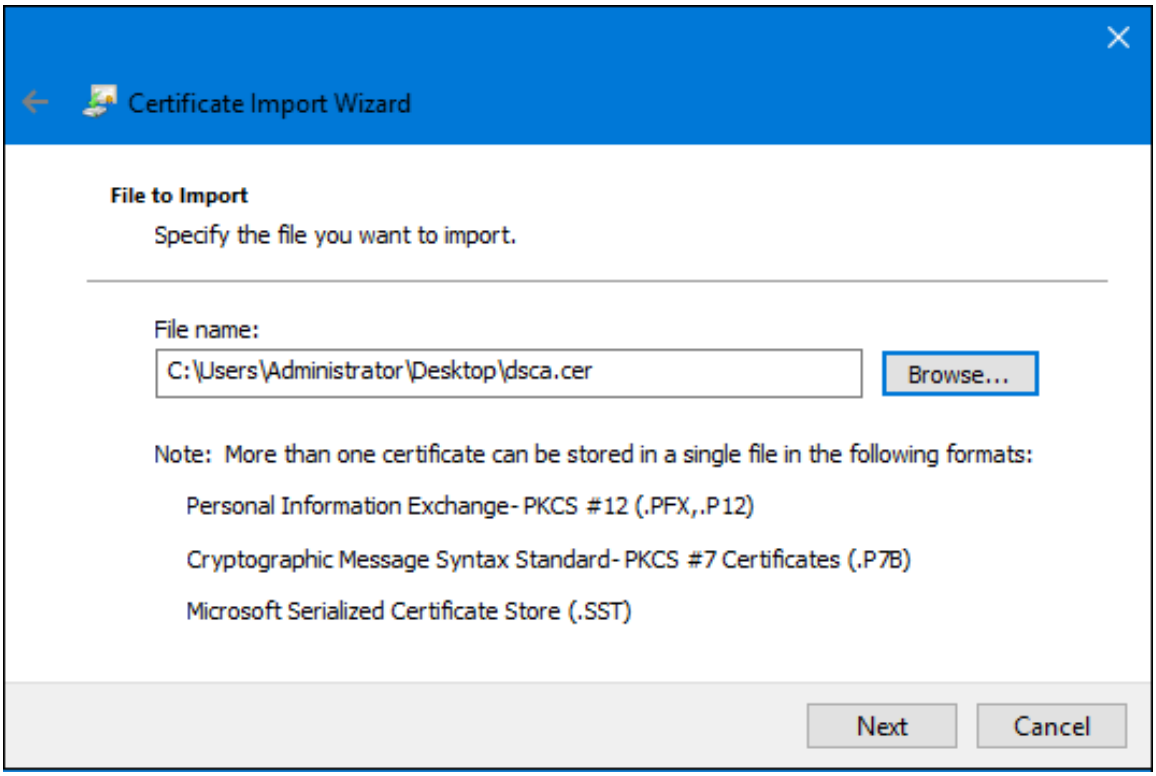
```
# cd /etc/dirsrv/slapd-instance_name/
# certutil -d . -L -n "CA certificate" -a > dsca.crt
```

5. Copy the exported certificate from the Directory Server to the Windows machine.
6. Import the CA certificate from Directory Server into AD.
   1. Open **Administrative Tools** and select the **Certificate Authority** item.
   2. Expand **Trusted Root Certification Authorities**.

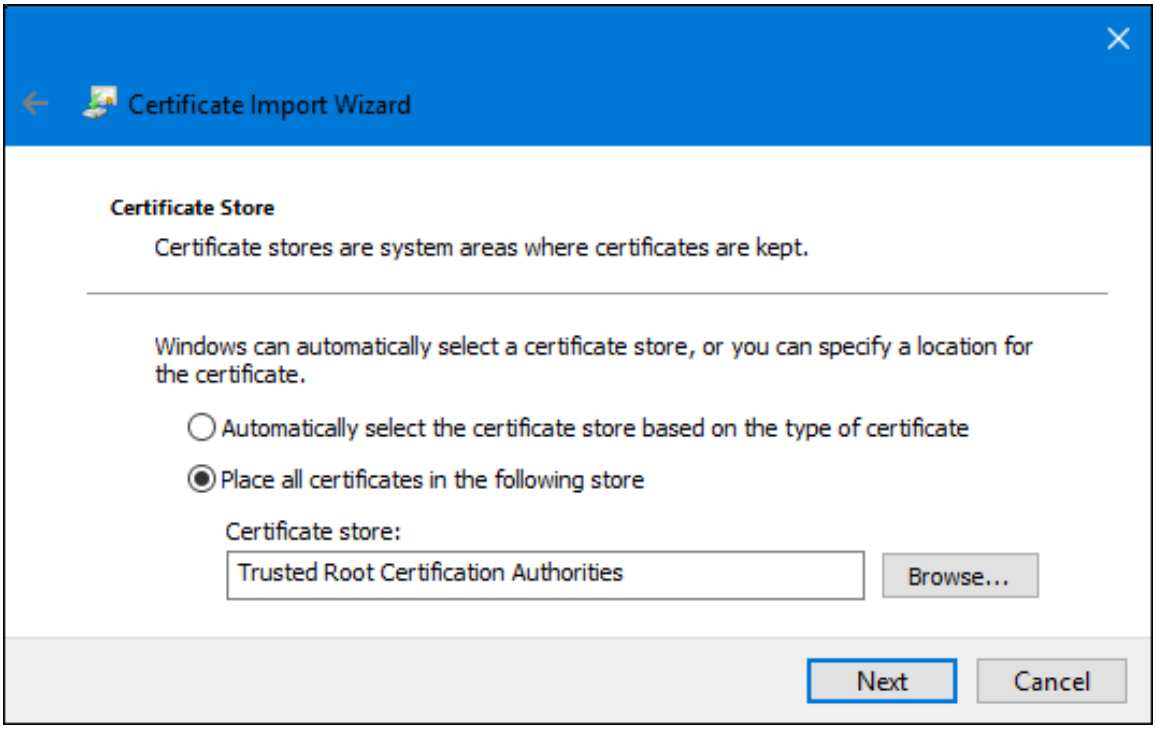
   3. Right-click the **Certificates** item and select **Import**.



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 TLS correctly, try searching AD over LDAPS.

16.4.3. Step 3: Select or Create the Synchronization Identity

There are two users used to configure Windows Synchronization:

- *An AD user, specified in the sync agreement.*

  The user specified in the sync agreement is the entity as whom the Directory Server binds to AD to send and receive updates. The AD 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 AD , 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 AD 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.

16.4.4. Step 4: Install the Password Sync Service

The steps to install the **Password Sync** service are described in the *Installing the Password Sync Service* section in the Red Hat Directory Server Installation Guide.

For a list of operating systems Red Hat supports running the **Password Sync** service on, see *Red Hat Directory Server Release Notes*.

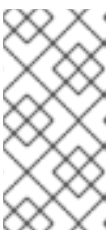
NOTE

The first attempt to synchronize passwords, which happens when the **Password Sync** application is installed, always fails because the CA certificate does not exist in **Password Sync**'s certificate database. Adding the CA certificate is part of the configuration step of the application.

16.4.5. Step 5: Configure the Password Sync Service

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

1. Enable TLS in Directory Server. For details, see Section 9.4.1, “Enabling TLS in Directory Server”.



NOTE

TLS is required for Password Sync to send passwords to Directory Server. The service will not send the passwords except over TLS 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 TLS is configured.

2. On the Directory Server, export the server certificate.

```
# certutil -d /usr/lib64/dirsrv/slapd-instance -L -n "CA certificate" -a > dsca.crt
```

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

4. Open a command prompt on the Windows machine, and open the **Password Sync** installation directory.

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

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

```
> certutil.exe -d . -N
```

6. Import the server 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
```

7. Verify that the CA certificate was correctly imported.

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

8. 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.

16.4.6. 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 15.6.1, “Configuring the Read-Write Replicas on the Supplier Servers”.

16.4.6.1. Setting up the Directory Server for Synchronization 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.

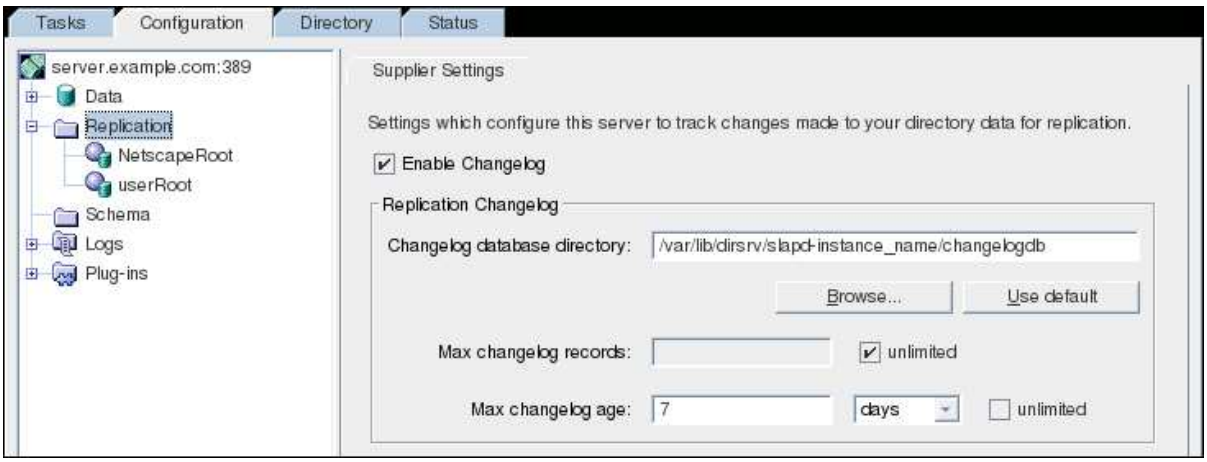
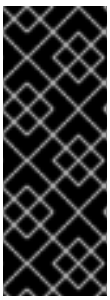


Figure 16.3. The Configuration tab

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 AD. The AD 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.

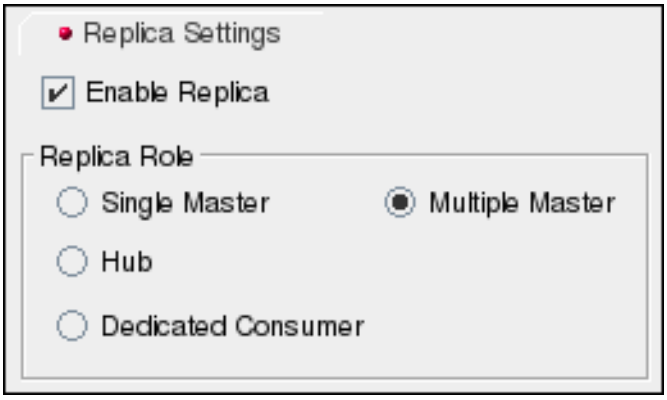


Figure 16.4. The Enable Replica check box

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 16.4.3, "Step 3: Select or Create the Synchronization Identity", this user must be on the Directory Server and must have the access right for the **userPassword** attribute of all users that are to be synchronized.

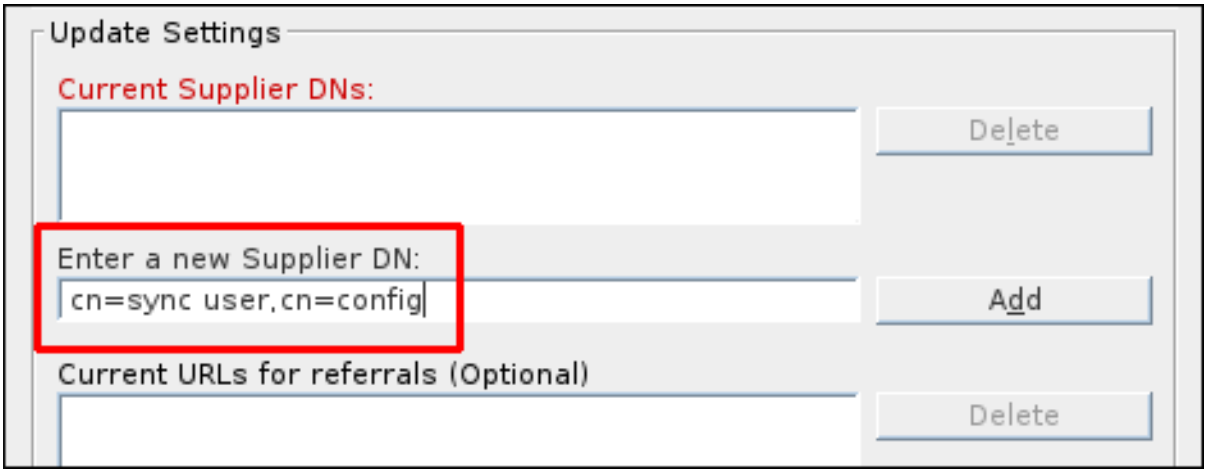
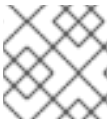


Figure 16.5. The Update Settings section

5. Save the replication settings for the database.



NOTE

For more information on replication settings, see Chapter 15, *Managing Replication*.

16.4.6.2. Setting up the Directory Server for Synchronization 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
nsslapd-changelogmaxage: 7d
```

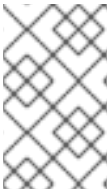
Then, create the supplier replica entry:

```
# 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, Command, and File Reference* and Section 15.2.1, "Configuring Suppliers from the Command Line".

16.4.7. Step 7: Create the Synchronization Agreement

Create the synchronization agreement.



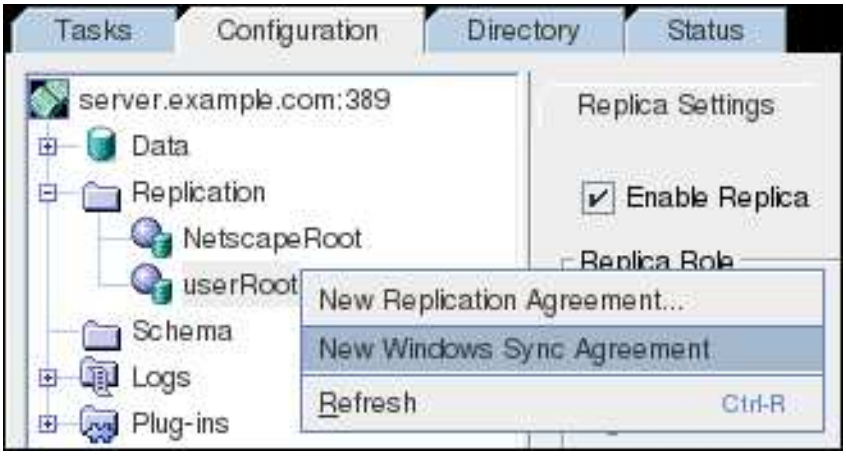
NOTE

If secure binds are required for simple password authentication (Section 19.11.1, "Requiring Secure Binds"), then any replication operations will fail unless they occur over a secure connection. Using a secure connection (LDAPS or StartTLS) is recommended, anyway.

16.4.7.1. Creating the Synchronization Agreement from the Console

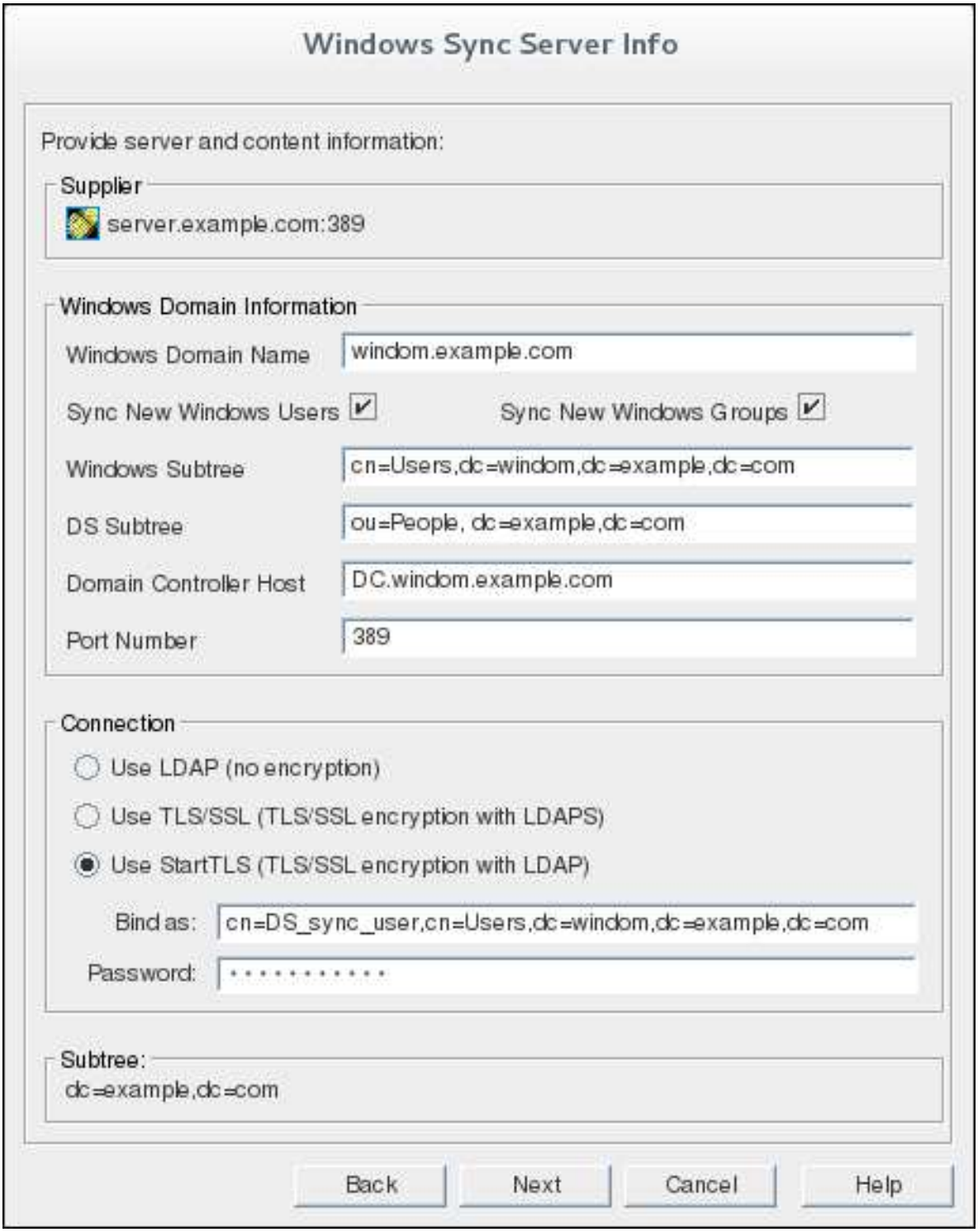
1. In the Directory Server Console, select the **Configuration** tab.
2. In the left-hand navigation tree, click **Replication**, then right-click on the database to sync. The default user database is **userRoot**, but additional databases are added as new suffixes are added to the Directory Server.

   Alternatively, highlight the database, and in the top tool bar, click **Object**.
3. Select **New Windows Synchronization 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 AD information in the **Windows Domain Information** area.



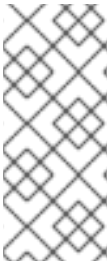
- The name of the Windows domain.
- What kinds of entries to synchronize; users and groups are synchronized independently. When a type of entry is chosen, then all of the entries of that type that are found in the Windows subtree are created in the Directory Server.
- The Windows and Directory Server 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 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 TLS, these peer servers must be able to trust each other's certificates.

Using either TLS or Start TLS is recommended for security reasons. TLS or Start TLS is required for synchronizing passwords because AD refuses to modify passwords unless the connection is TLS-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 AD.
8. Save the sync agreement.



NOTE

By default, Windows Synchronization polls the AD 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 16.12.2, "Adding and Editing the Synchronization Agreement in the Command Line".

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

16.4.7.2. Creating the Synchronization 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=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: add
objectclass: top
objectclass: nsDSWindowsReplicationAgreement
cn: replication_agreement_name
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
```

For descriptions of the parameters used in the example and other attributes you can set, see the *Red Hat Directory Server Configuration, Command, and File Reference* .

16.4.8. 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. AD entries which are synchronized 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 synchronized over at the next incremental update.

Configuring Directory Server user entries for synchronization is described in Section 16.5.3, "Configuring User Synchronization for Directory Server Users", and configuring Directory Server group entries for synchronization is described in Section 16.6.4, "Configuring Group Synchronization for Directory Server Groups".

16.4.9. Step 9: Begin Synchronization

After the synchronization agreement has been created, start the synchronization.

Starting the Synchronization Using the Command Line

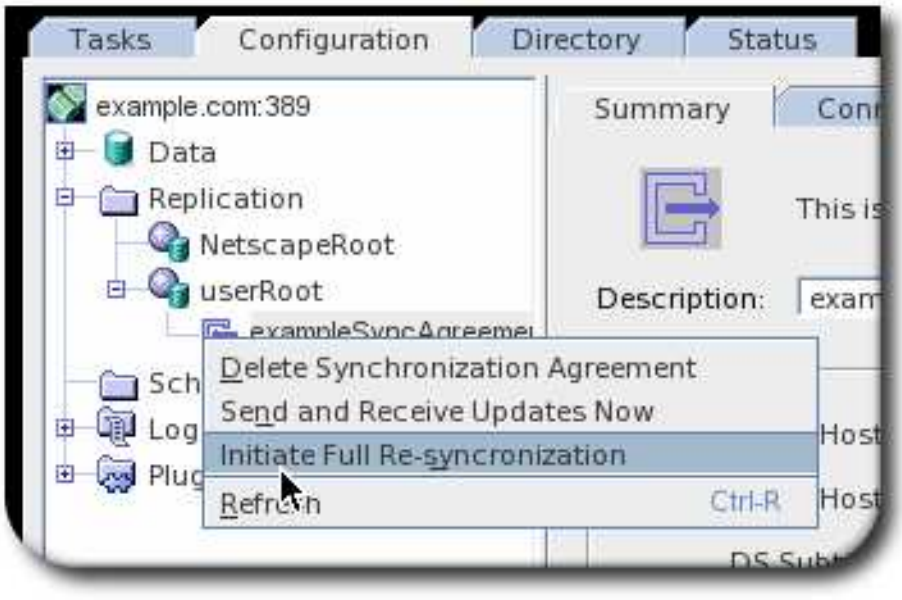
To start synchronization using the command line:

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

dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
replace: nsds5beginreplicarefresh
nsds5beginreplicarefresh: start
```

When the initialization is complete, Directory Server automatically removes the ***nsds5BeginReplicaRefresh*** attribute from the replication agreement entry.

Starting the Synchronization Using the Console

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.

16.5. SYNCHRONIZING USERS

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

- Users in the Active Directory domain are synchronized 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 synchronized 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 the Directory Server Windows Synchronization plug-in to write the entry over to the Active Directory server.

  New or modified user entries with the **ntUser** object class added are created and synchronized 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 synchronized 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 synchronized 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 synchronized on Active Directory.

16.5.1. 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 16.2, "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 16.1, "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 16.5.2, "User Schema Differences between Red Hat Directory Server and Active Directory".

Table 16.1. User Schema Mapped between Directory Server and Active Directory

Directory Server	Active Directory
cn[a]	name
ntUserDomainId	sAMAccountName
ntUserHomeDir	homeDirectory
ntUserScriptPath	scriptPath
ntUserLastLogon	lastLogon

Directory Server	Active Directory
ntUserLastLogoff	lastLogoff
ntUserAcctExpires	accountExpires
ntUserCodePage	codePage
ntUserLogonHours	logonHours
ntUserMaxStorage	maxStorage
ntUserProfile	profilePath
ntUserParms	userParameters
ntUserWorkstations	userWorkstations

[a] The ***cn*** is treated differently than other synchronized 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 16.2. 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 synchronized 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.

16.5.2. 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.

16.5.2.1. 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.

16.5.2.2. Password Policies

Both Active Directory and Directory Server can enforce password policies such as password minimum length or maximum age. Windows Synchronization 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 synchronized to the other system. The default password syntax setting on Directory Server mimics the default password complexity rules that Active Directory enforces.

16.5.2.3. 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 Synchronization 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 synchronized 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.

16.5.2.4. 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.

16.5.3. Configuring User Synchronization 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.

16.5.3.1. Configuring User Synchronization in the Console

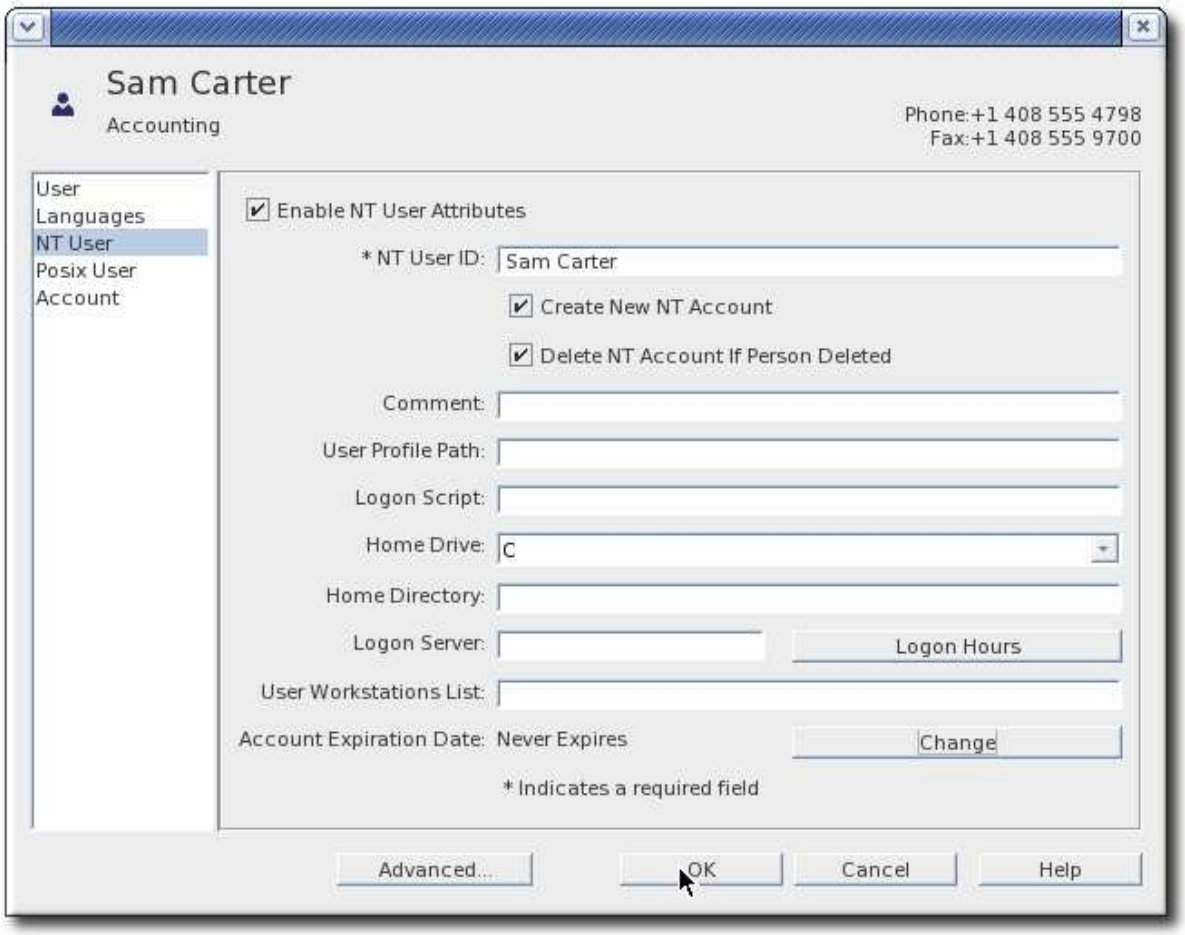
1. In the Directory Server Console, select the **Directory** tab.

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.

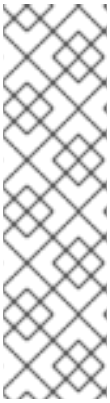
3. On the left side of the **Property Editor**, click the **NT User** link.

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 3.1.4, "Updating a Directory Entry" .



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 synchronized 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.

16.5.3.2. Configuring User Synchronization 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, using the **ldapmodify** utility:

```
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 16.5.1, "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 10 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 synchronized 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.

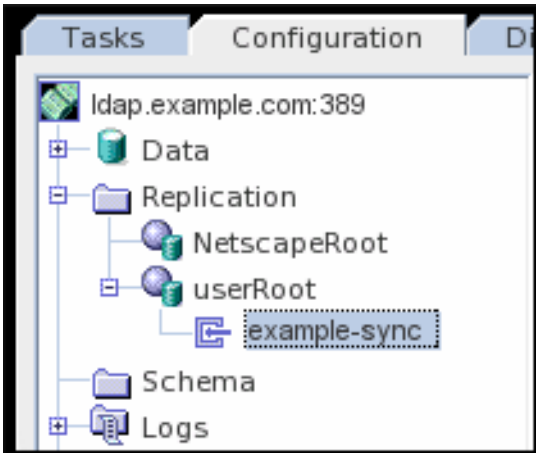
So, to make the user active on the Active Directory domain, reset the user's password.

16.5.4. Configuring User Synchronization for Active Directory Users

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

16.5.4.1. Configuring User Synchronization 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.

16.5.4.2. Configuring User Synchronization 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** using **ldapmodify**:

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

dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
replace: nsds7NewWinUserSyncEnabled
nsds7NewWinUserSyncEnabled: on
```

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

16.6. SYNCHRONIZING GROUPS

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

- Groups in the Active Directory domain are synchronized 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 synchronized 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 Windows Synchronization to write the entry over to the Active Directory server.

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

IMPORTANT

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

This could create a problem when an application or service tries to do a modify operation on all members in a group on the Active Directory server, if 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, builtin, universal/security, global/distribution, domain local/distribution, or universal/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 synchronized.

16.6.1. 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:

- **-2147483646** for global/security (the default)

- **-2147483644** for domain local/security

- **-2147483643** for builtin

- **-2147483640** for universal/security

- **2** for global/distribution
- **4** for domain local/distribution
- **8** for universal/distribution

16.6.2. 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 16.4, "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 16.3, "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 16.6.3, "Group Schema Differences between Red Hat Directory Server and Active Directory".

Table 16.3. Group Entry Attribute Mapping between Directory Server and Active Directory

Directory Server	Active Directory
cn	name
ntUserDomainID	name
ntGroupType	groupType
uniqueMember member	Member[a]

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

Table 16.4. Group Entry Attributes That Are the Same between Directory Server and Active Directory

cn	o
description	ou
l	seeAlso
mail	

16.6.3. 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.

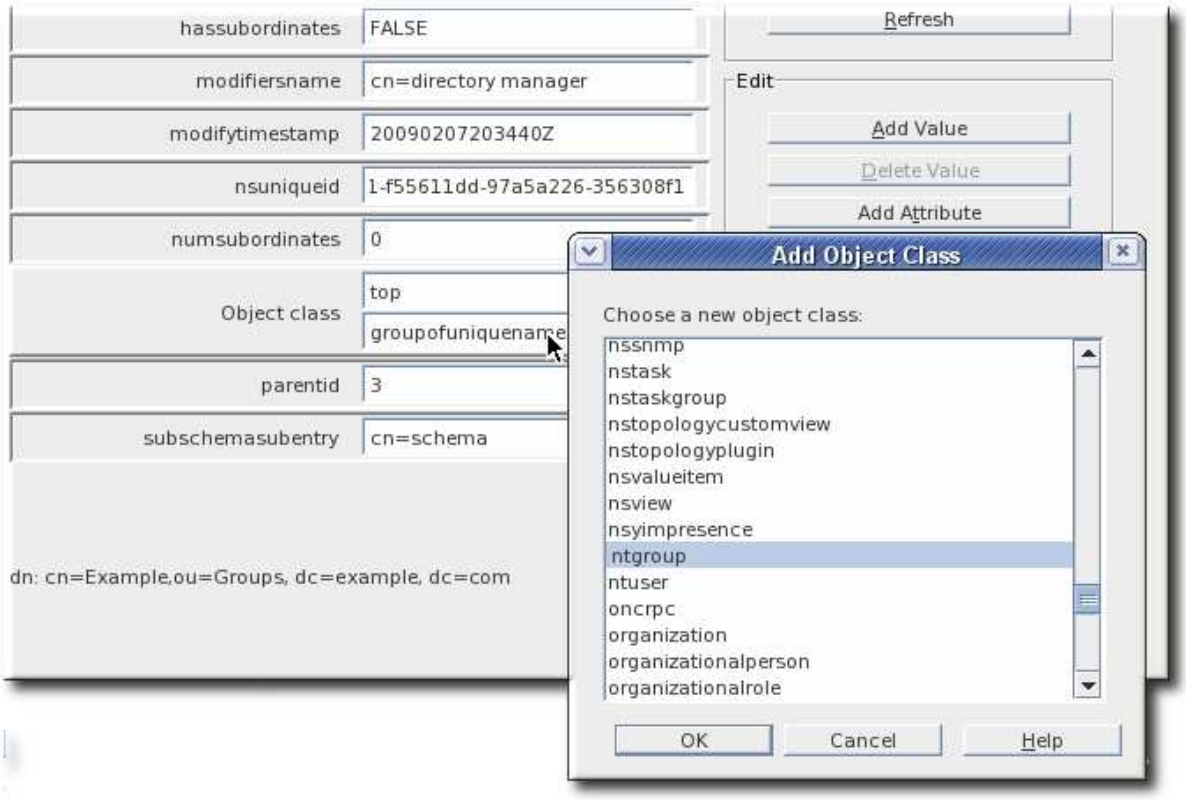
Nested groups (where a group contains another group as a member) are supported and for Windows Synchronization are synchronized. However, Active Directory imposes certain constraints as to the composition of nested groups. For example, a global group is not allowed to contain a domain local group as 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.

16.6.4. Configuring Group Synchronization 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.

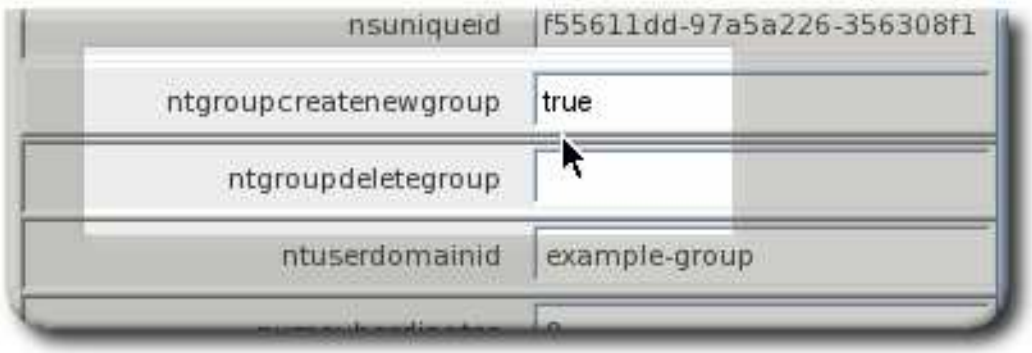
16.6.4.1. Configuring Group Synchronization in the Console

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 16.6.2, "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:-2147483646**).

16.6.4.2. Configuring Group Synchronization 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 add the ***ntGroupType*** attribute. If this attribute is not specified, then the group is automatically added as a global security group (**ntGroupType:-2147483646**).

For example, using **ldapmodify**:

```
# 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
```

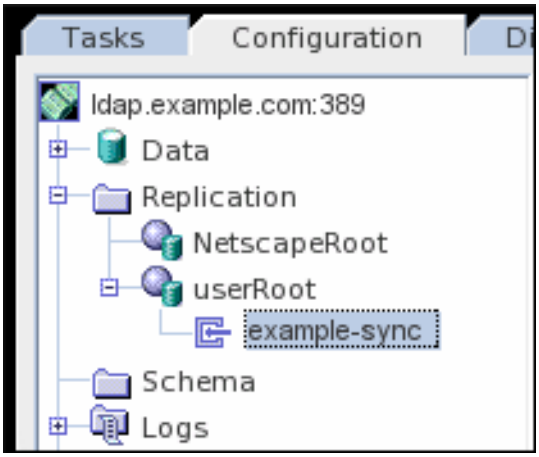
Many additional Windows and group attributes can be added to the entry. All of the schema which is synchronized is listed in Section 16.6.2, "Group Attributes Synchronized between Directory Server and Active Directory". Windows-specific attributes, belonging to the **ntGroup** object class, are described in more detail in the *Red Hat Directory Server 10 Configuration, Command, and File Reference* .

16.6.5. Configuring Group Synchronization for Active Directory Groups

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

16.6.5.1. Configuring Group Synchronization 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.

16.6.5.2. Configuring Group Synchronization 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**. Using **ldapmodify**:

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

dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
replace: nsds7NewWinGroupSyncEnabled
nsds7NewWinGroupSyncEnabled: on
```

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

16.7. CONFIGURING UNI-DIRECTIONAL SYNCHRONIZATION

As Figure 16.1, "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[2] 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.

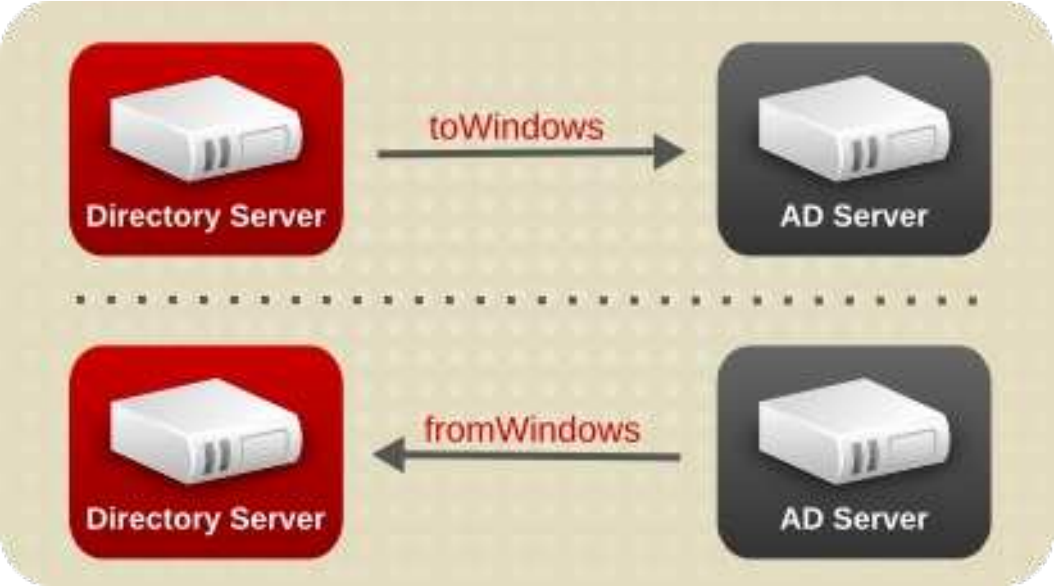


Figure 16.6. Uni-Directional Synchronization

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 Active Directory to Directory Server synchronization, 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 to Active Directory synchronization, 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 16.4.7, "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. Using **ldapmodify**:

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

dn: cn=replication_agreement_name,cn=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 than 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*synchronized server. Access controls for Directory Server are covered in Chapter 18, *Managing Access Control*. For Active Directory, see the appropriate Windows documentation.

Uni-directional sync does not affect password synchronization. Even when ***oneWaySync*** is set to ***toWindows***, after updating a password on the Active Directory server, the password is sent to the Directory Server.

16.8. CONFIGURING MULTIPLE SUBTREES AND FILTERS IN WINDOWS SYNCHRONIZATION

Windows Synchronization is designed to synchronize between multiple pairs of subtrees on the Directory Server (DS) and Active Directory (AD). By using filters, only specified entries under a subtree are synchronized.

Multiple Subtrees in Windows Synchronization

To synchronize among multiple subtree pairs, configure the Directory Server and the Active Directory subtrees in the **winSyncSubtreePair** parameter in the Windows sync agreement. Use **ldapmodify** to set multiple subtrees as follows:

```
changetype: modify
add: winSyncSubtreePair
winSyncSubtreePair: ou=OU1,dc=DSexample,dc=com:ou=OU1,DC=ADexample,DC=com
```

If **winSyncSubtreePair** is not set, the **nsds7WindowsReplicaSubtree** AD subtree parameter and the **nsds7DirectoryReplicaSubtree** DS subtree parameter are used for the synchronization target checks instead. Otherwise, these two parameters are ignored.

Filters in Windows Synchronization

You can set a filter that selects data to be synchronized in the following parameters:

- **winSyncWindowsFilter** sets an additional filter on the Active Directory server,

- **winSyncDirectoryFilter** parameter sets an additional filter on the Directory Server.

In the following example, **ldapmodify** is used to synchronize entries whose CN contains **user** or **group**:

```
changetype: modify
add: winSyncWindowsFilter
winSyncWindowsFilter: (|(cn=*user*)(cn=*group*))
-
add: winSyncDirectoryFilter
winSyncDirectoryFilter: (|(uid=*user*)(cn=*group*))
```

16.9. 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 16.5.1, "User Attributes Synchronized between Directory Server and Active Directory" and Section 16.6.2, "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 synchronized over, they have ***ntUser*** and ***ntGroup*** attributes automatically added which identify them as Windows accounts, but no POSIX attributes are synchronized 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.

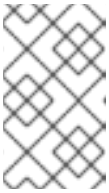
16.9.1. Enabling POSIX Attribute Synchronization

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.

16.9.2. Changing Posix Group Attribute Synchronization Settings

There are multiple plug-in attributes that can be set to control how the POSIX group attributes and group members are synchronized 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.

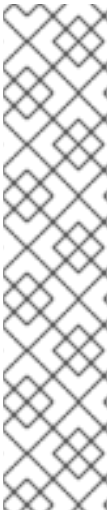
16.10. DELETING AND RESURRECTING ENTRIES

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

16.10.1. Deleting Entries

All changes on an Active Directory peers are always synchronized 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 synchronized 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.

16.10.2. 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 synchronized 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.

16.11. 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.

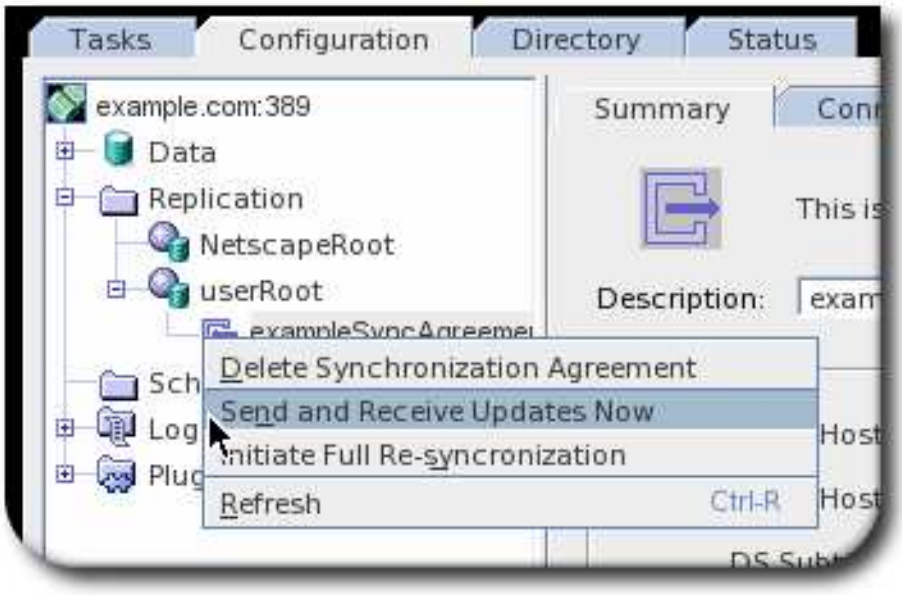
A sync update can be triggered manually. It is also possible to do a full resynchronization, which sends and pulls every entry in the Directory Server and Active Directory as if it were new. A full resynchronization includes existing Directory Server entries which may not have previously been synchronized.

16.11.1. Performing a Manual Incremental Synchronization

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.



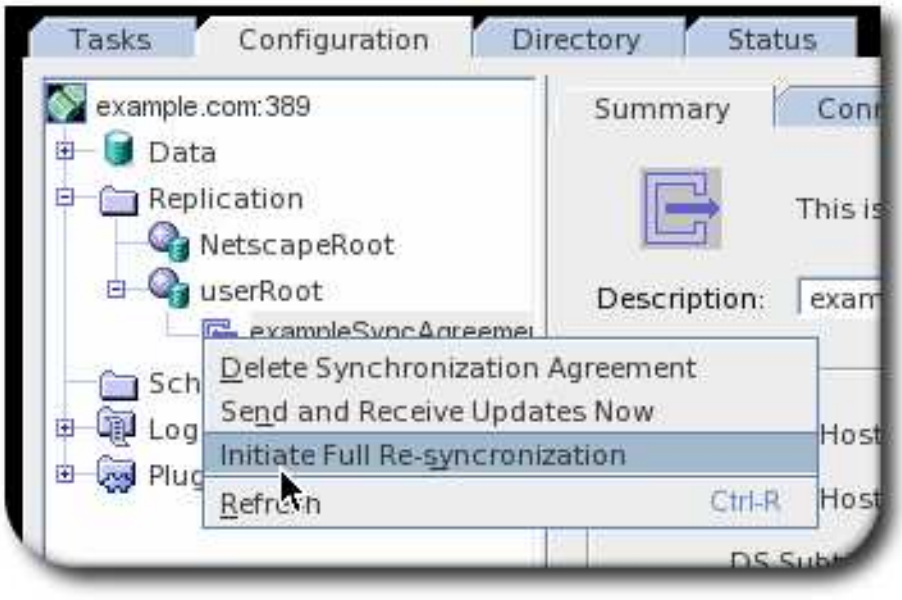
16.11.2. Performing a Full Synchronization

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.

16.11.2.1. Performing a Full Synchronization using the Console

To perform a full synchronization:

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.

16.11.2.2. Performing a Full Synchronization using the Command Line

To start a full synchronization using the command line, add the ***nsDS5BeginReplicaRefresh*** attribute with the **start** value to the synchronization agreement.

For example, to start a full synchronization for the **Example** agreement:

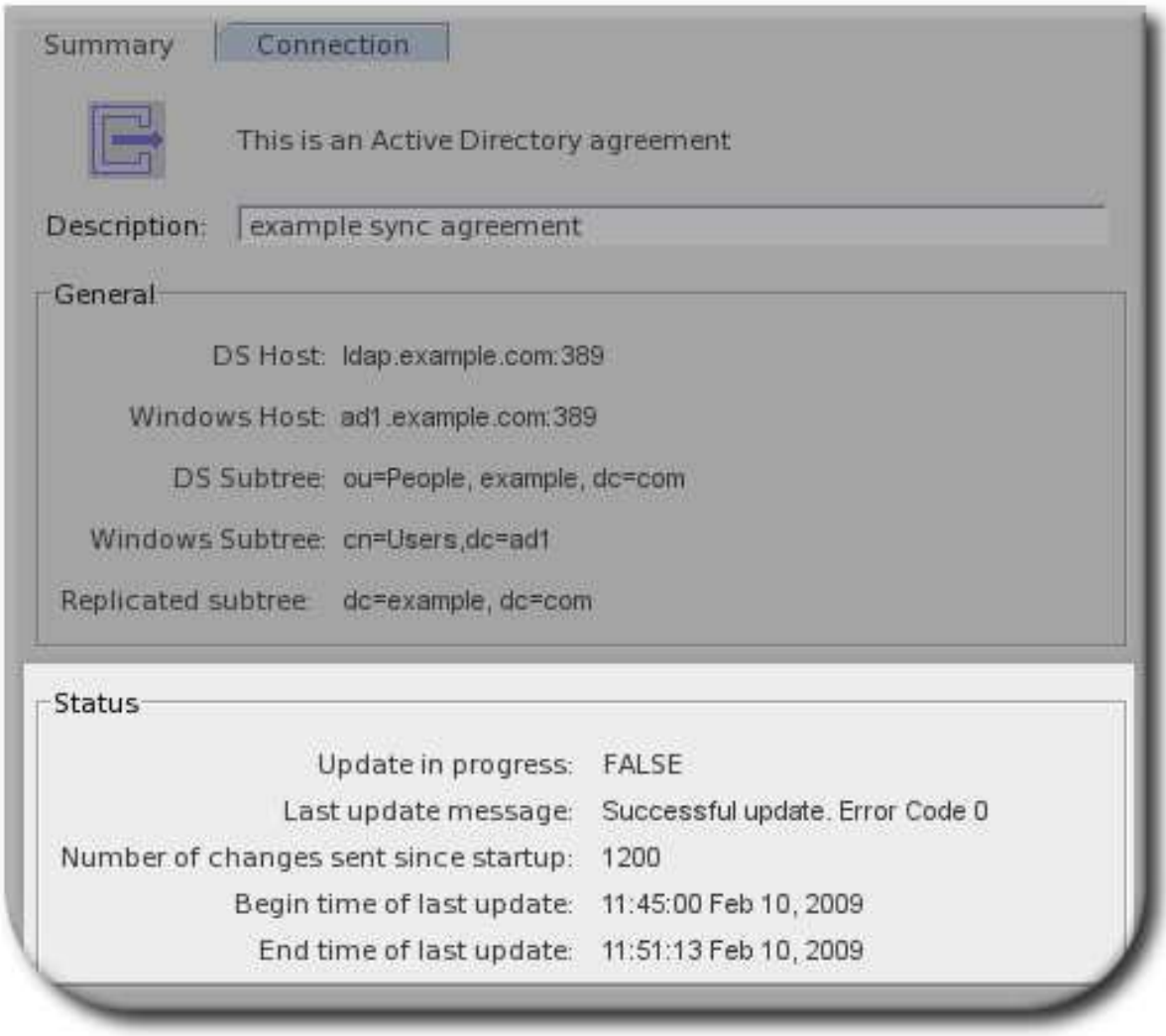
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
add: nsDS5BeginReplicaRefresh
nsDS5BeginReplicaRefresh: start
```

After the synchronization, Directory Server automatically removes the ***nsDS5BeginReplicaRefresh*** attribute from the agreement entry.

16.11.3. 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.



16.12. MODIFYING THE SYNCHRONIZATION 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.

16.12.1. Editing the Synchronization 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 synchronized. All of the synchronization agreements are listed below the database. Double-click the sync agreement to open it in the main window.

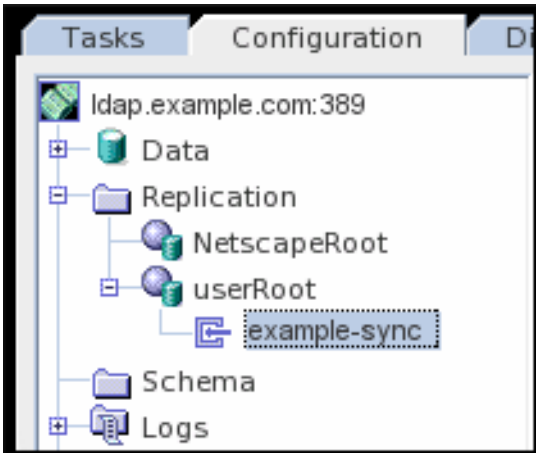


Figure 16.7. Selecting the Synchronization Agreement

3. Click the **Connection** tab.

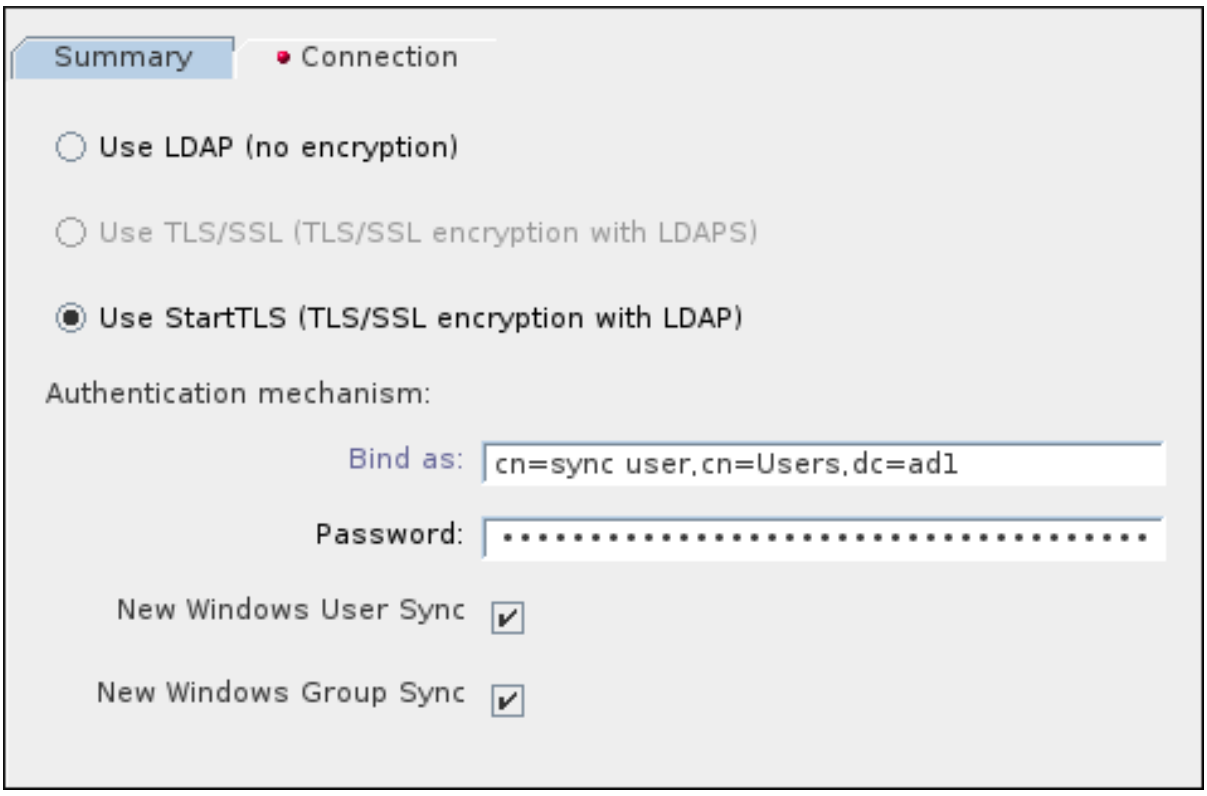


Figure 16.8. The Connection tab

There are three areas of information that can be edited.

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

16.12.2. Adding and Editing the Synchronization 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 described in the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

16.12.2.1. Creating a Basic Synchronization 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***)
- 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**), TLS (**SSL**), or StartTLS (**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, using **ldapmodify**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: add
objectclass: top
objectclass: nsDSWindowsReplicationAgreement
cn: replication_agreement_name
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=Users,dc=ad1
nsDS5ReplicaCredentials: {DES}ffGad646dT0nnsT8nJOaMA==
nsDS5ReplicaTransportInfo: TLS
nsds7NewWinUserSyncEnabled: on
nsds7NewWinGroupSyncEnabled: on
```

To synchronize among multiple subtree pairs, see Section 16.8, "Configuring Multiple Subtrees and Filters in Windows Synchronization".

16.12.2.2. Setting Synchronization 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.

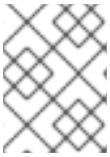
By default, the Directory Server update schedule is to always be in sync. The Active Directory interval is to poll the Active Directory every five minutes.

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
```

16.12.2.3. Changing Synchronization 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=Users,dc=ad1
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 TLS connections, since that requires changing between standard and insecure ports.

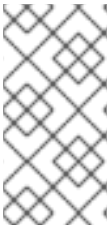
To change to or from TLS, delete the sync agreement and add it again with the updated port number and new transport information.

16.12.2.4. Handling Entries That Move Out of the Synchronized 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 synchronized 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 synchronized.

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 synchronized 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.

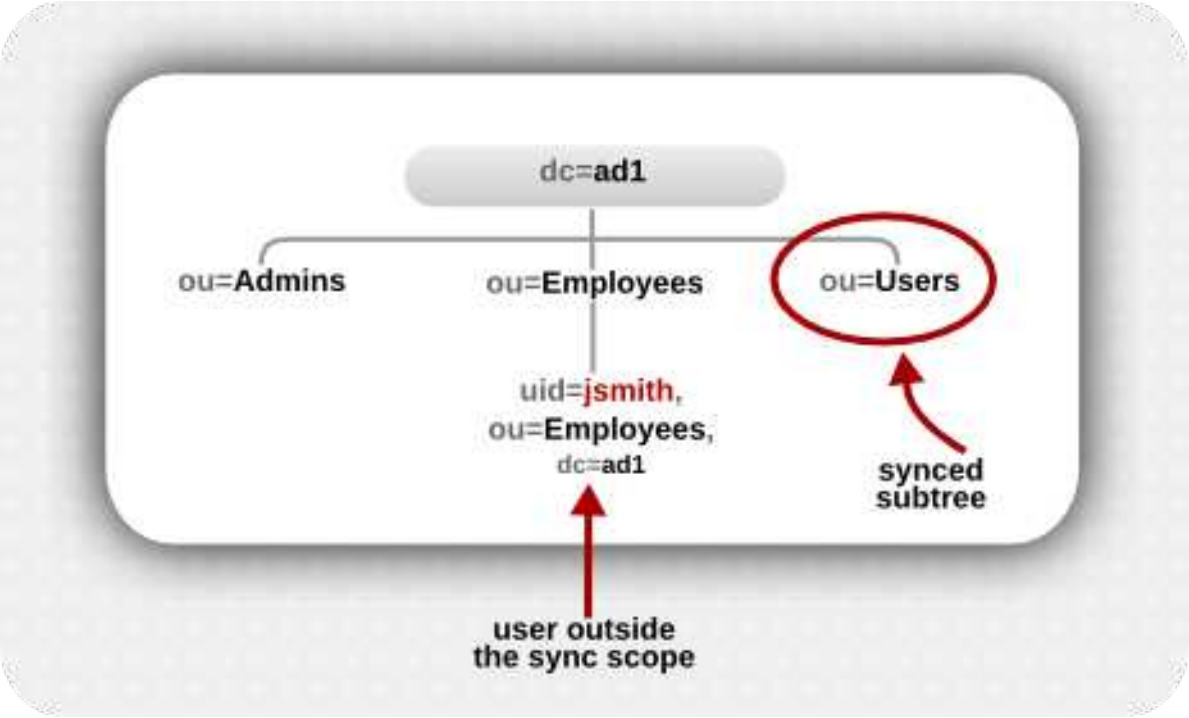


Figure 16.9. Active Directory Tree

For 7.x and 8.x versions of Directory Server, synchronization simply ignored that user, since it was outside the synchronized 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 synchronized subtree are *intentionally* moved outside the synchronized subtree – essentially, a rename operation. The assumption then was that the "corresponding" Directory Server entry should be deleted.

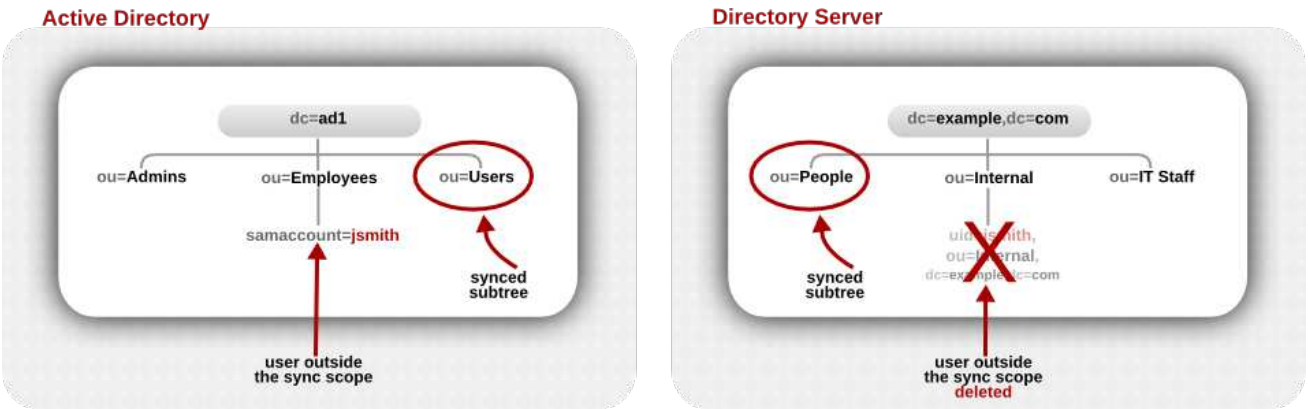


Figure 16.10. Active Directory and Directory Server Trees Compared

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

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

- **none** takes no action, so if a synchronized Directory Server entry exists, it may be synchronized over to or create an Active Directory entry *within* scope. If no synchronized Directory Server entry exists, nothing happens at all (this is the default behavior in the Directory Server version 9.1 and later).
- **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 synchronized 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. Using **ldapmodify**:

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

dn: cn=replication_agreement_name,cn=replica,cn="dc=example,dc=com",cn=mapping
tree,cn=config
changetype: modify
add: winSyncMoveAction
winSyncMoveAction: unsync
```

16.13. 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.

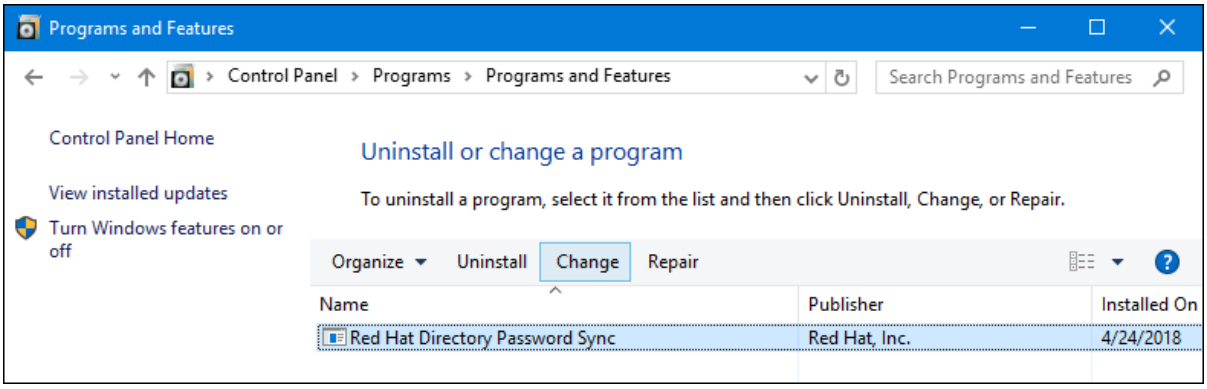
The service synchronizes password changes made on Active Directory with the corresponding entries' passwords on the Directory Server. Like any Windows service, it can be modified, started and stopped, and uninstalled, depending on how synchronization between Directory Server and Active Directory changes.

16.13.1. Modifying Password Sync

To reconfigure **Password Sync**:

1. Open **Control Panel** and select **Programs and Features**.

2. Select the **Red Hat Directory Password Sync** entry, and 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.

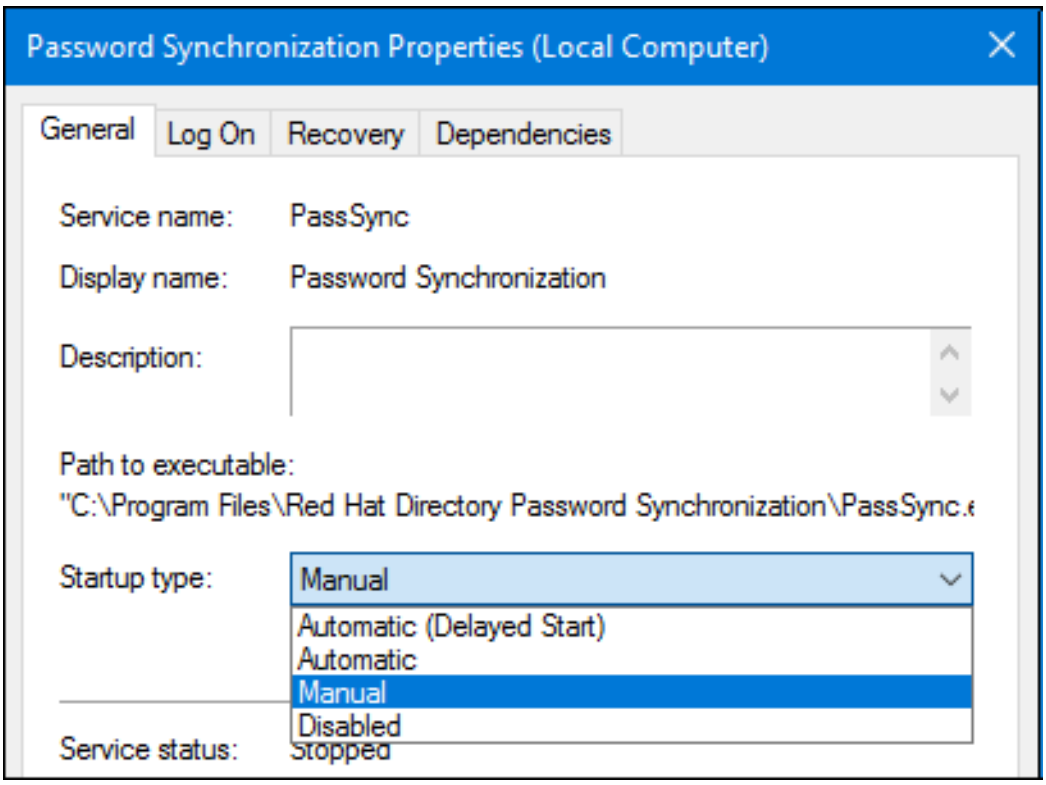
16.13.2. Starting and Stopping the Password Sync Service

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. Open the **Services** application.

2. Double-click the **Password Synchronization** service.

3. Select the **Manual** radio button, and click **OK**.

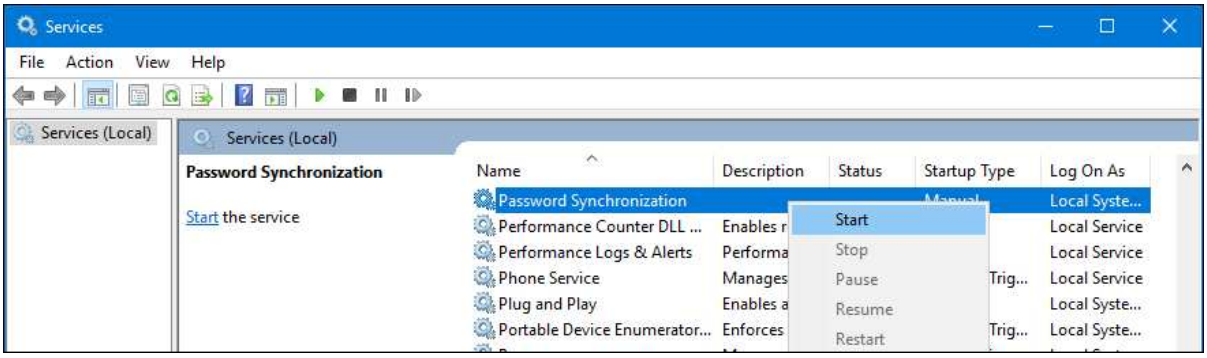


To start and stop **Password Sync**:

1. Open the **Services** application.

2. Right-click the **Password Synchronization** service.

3. Select **Stop**, **Start**, or **Restart**, and click **OK**

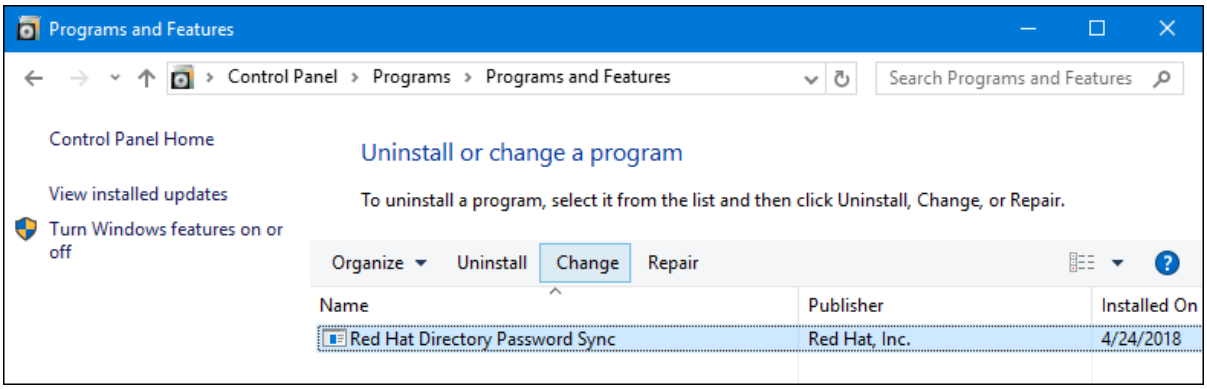


Changed passwords are captured even if the **Password Synchronization** service is not running. If **Password Synchronization** is restarted, the password changes are sent to Directory Server at the next synchronization.

16.13.3. Uninstalling Password Sync Service

1. Open **Control Panel** and select **Programs and Features**.

2. Select the **Red Hat Directory Password Sync** entry, and click the **Uninstall** button.



3. If TLS 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 manually.

16.13.4. Upgrading Password Sync

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

16.14. 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 TLS 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.

---

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

CHAPTER 17. SETTING UP CONTENT SYNCHRONIZATION

Using the **Content Synchronization** plug-in, Directory Server supports the **SyncRepl** protocol according to RFC 4533. This protocol enables LDAP servers and clients to use Red Hat Directory Server as a source to synchronize their local database with the changing content of Directory Server.

To use the **SyncRepl** protocol:

- Enable the **Content Synchronization** plug-in in Directory Server and optionally create a new user which the client will use to bind to Directory Server. The account must have permissions to read the content in the directory.
- Configure the client. For example, set the search base for a subtree to synchronize. For further details, see your client's documentation.

Before clients are able to connect to Directory Server, set up the **Content Synchronization** plug-in:

1. The **Content Synchronization** plug-in requires the **Retro Changelog** plug-in to log the ***nsuniqueid*** attribute:

   a. To verify if the retro changelog is already enabled, enter:

   ```
   # ldapsearch -D "cn=Directory Manager" -W -x -b \
       'cn=Retro Changelog Plugin,cn=plugins,cn=config' nsslapd-pluginEnabled
   ...
   dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
   nsslapd-pluginEnabled: off
   ```

   If the ***nsslapd-pluginEnabled*** attribute is set to **off**, the retro changelog is disabled. To enable, see Section 15.23.1, “Enabling the Retro Changelog Plug-in” .

   b. Add the ***nsuniqueid*** attribute to retro changelog plug-in configuration:

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

   dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
   changetype: add
   add: nsslapd-attribute
   nsslapd-attribute: nsuniqueid:targetUniqueId
   ```

   c. Optionally, apply the following recommendations for improved performance:

      i. Set maximum validity for entries in the retro change log. For example, to set 2 days (**2d**):

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

      dn: cn=changelog5,cn=config
      changetype: modify
      replace: nsslapd-changelogmaxage
      nsslapd-changelogmaxage: 2d
      ```

      ii. If you know which back end or subtree clients access to synchronize data, limit the scope of the **Retro Changelog** plug-in. For example, to exclude the ***cn=demo,dc=example,dc=com*** subtree, enter:

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

dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-exclude-suffix
nsslapd-exclude-suffix: cn=demo,dc=example,dc=com
```

2. Enable the **Content Synchronization** plug-in:
   - Using the command line:

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

dn: cn=Retro Changelog Plugin,cn=plugins,cn=config
changetype: modify
replace: nsslapd-pluginEnabled
nsslapd-pluginEnabled: on
```

   - Using the Directory Server Console: See Section 1.9.2.2, “Enabling Plug-ins in the Directory Server Console”.

3. Using the defaults, Directory Server creates an access control instruction (ACI) in the ***oid=1.3.6.1.4.1.4203.1.9.1.1,cn=features,cn=config*** entry that enables all users to use the **SyncRepl** protocol:

```
aci: (targetattr != "aci")(version 3.0; acl "Sync Request Control";
  allow( read, search ) userdn = "ldap:///all";)
```

   Optionally, update the ACI to limit using the **SyncRepl** control. For further details about ACIs, see Section 18.13, “Defining Bind Rules”.

4. Restart Directory Server:

```
# systemctl restart dirsrv@instance_name
```

Clients are now able to synchronize data with Directory Server using the **SyncRepl** protocol.

CHAPTER 18. MANAGING ACCESS CONTROL

This chapter describes how you use Access Control Instructions (ACI) in Red Hat Directory Server to manage access to entries.

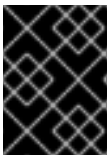
18.1. ACCESS CONTROL PRINCIPLES

When Directory Server receives a request, it uses the authentication information provided by the user in the bind operation and the ACIs defined in the directory to allow or deny access to the requested entry or attribute. The server can allow or deny permissions for actions, such as **read**, **write**, **search**, and **compare**. The permission level granted to a user depends on the authentication information provided.

Access control in Directory Server enables you to set precise rules on when the ACIs are applicable:

- For the entire directory, a subtree, or specific entries
- For a specific user, all users belonging to a specific group or role, or all users in the directory
- For a specific location, such as an IP address, an IP range, or a DNS name.

  Note that load balancers can affect location-specific rules.



IMPORTANT

Complex ACIs are difficult to read and understand. Instead of one complex ACI in most cases, it is better to write multiple simple rules to achieve the same effect.

18.2. ACI PLACEMENT

Directory Server stores ACIs in the multi-valued ***aci*** operational attribute in directory entries. To set an ACI, add the ***aci*** to the corresponding directory entry. Directory Server applies the ACIs:

- Only to the entry that contains the ACI, if it does not have any child entries.

  For example, if a client requires access to the **uid=*user_name*,ou=People,dc=example,dc=com** object, and an ACI is only set on **dc=example,dc=com** and not on any child entries, only this ACI is applied.

- To the entry that contains the ACI and to all entries below it, if it has child entries. 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.

  For example, ACIs are set on the **dc=example,dc=com** and the **ou=People,dc=example,dc=com** entry. If a client wants to access the **uid=*user_name*,ou=People,dc=example,dc=com** object, which has no ACI set, Directory Server first validates the ACI on the **dc=example,dc=com** entry. If this ACI grants access, Directory Server then verifies the ACI on **ou=People,dc=example,dc=com**. If this ACI successfully authorizes the client, they can access the object.

NOTE

ACIs set in the **rootDSE** entry apply only to this entry.

An ACI created on an entry can be set not to apply directly to that entry but rather to some or all of the entries in the subtree below. The advantage of this approach is that general ACIs can be placed higher in the directory tree to have effect on entries 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.

NOTE

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 to leaf entries as closely as possible.

18.3. ACI STRUCTURE

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

(*target_rule*) (version 3.0; acl "*ACL_name*"; *permission_rule bind_rules*;)

- ***target_rule*** specifies the entry, attributes, or set of entries and attributes for which to control access. For details, see Section 18.11, “Defining Targets”.
- ***version 3.0*** is a required string which identifies the ACI version.
- ***permission_rule*** sets what rights, such as **read** or **write**, are allowed or denied. For details, see Section 18.12, “Defining Permissions”.
- ***bind_rules*** specifies which rules must match during the bind to allow or deny access. For details, see Section 18.13, “Defining Bind Rules”.

NOTE

The permission and the bind rule pair are called an access control rule.

To efficiently set multiple access controls for a given target, you can set multiple access control rules for each target:

(*target_rule*)(version 3.0; acl "*ACL_name*"; *permission_rule bind_rules*; *permission_rule bind_rules*; ... ;)

18.4. ACI EVALUATION

To evaluate the access rights to a particular entry, the server creates 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 databases for a particular instance but not across different instances.

Directory Server evaluates this list of ACIs 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.

In Directory Server, the **deny** permission in ACIs take precedence over the **allow** permission. For example, if you deny write permission at the directory's root level, none of the users can write to the directory, regardless if an other ACI grants this permission. To grant a specific user write permissions to

the directory, you have to add an exception to the original denying rule to allow the user to write in that directory.

NOTE

For improved ACIs, use fine-grained **allow** rules instead of **deny** rules.

18.5. LIMITATIONS OF ACIS

When you set ACIs, the following restrictions apply:

- If your directory database is distributed over multiple servers, the following restrictions apply to the keywords you can use in ACIs:
  - ACIs depending on group entries using the **groupdn** keyword must be located on the same server as the group entry.

    If the group is dynamic, all members of the group must have an entry on the server. Member entries of static groups can be located on the remote server.
  - ACIs depending on role definitions using the **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 by, for example, using the **userattr** keyword. In this case, access is evaluated normally even if the bind user does not have an entry on the server that stores the ACI.

  For further details, see Section 2.3.6, “Database Links and Access Control Evaluation” .
- You cannot use virtual attributes, such as Class of Service (CoS) attributes, in the following ACI keywords:
  - **targetfilter**
  - **targattrfilters**
  - **userattr**

  For details, see Chapter 8, *Organizing and Grouping Entries*.
- Access control rules are evaluated only on the local server. For example, if you specify the host name of a server in LDAP URLs in ACI keywords, the URL will be ignored.

18.6. HOW DIRECTORY SERVER HANDLES ACIS IN A REPLICATION TOPOLOGY

ACIs are stored in ***aci*** attributes of entries. Therefore, if an entry containing ACIs is part of a replicated database, the ACIs are replicated.

ACIs are always evaluated on server that resolves the incoming LDAP requests. 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.

18.7. DISPLAYING ACIS

This section describes how to display ACIs.

18.7.1. Displaying ACIs Using the Command Line

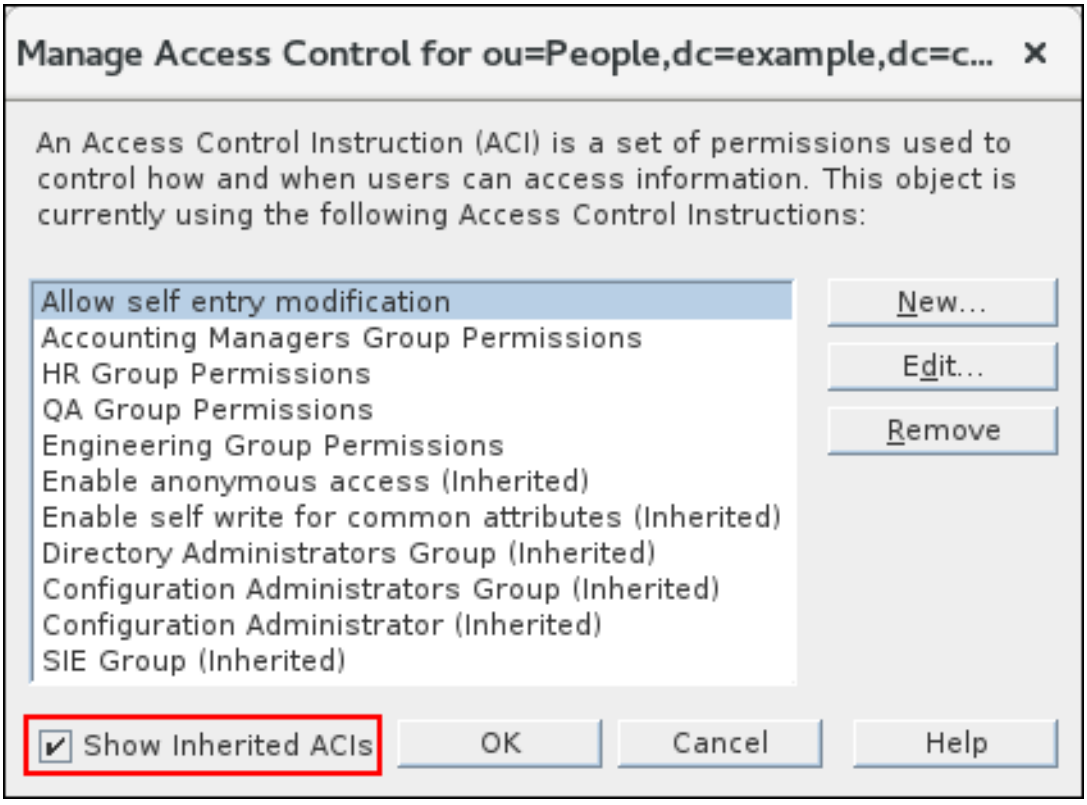
Use the **ldapsearch** utility to display ACI using the command line. For example, to display the ACIs set on **dc=example,dc=com** and sub-entries:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 -h server.example.com -x \
    -b "dc=example,dc=com" -s sub '(aci=*)' aci
```

18.7.2. Displaying ACIs Using the Console

To display ACIs using the Console:

1. Open the Directory Server Console.
2. On the **Directory** tab, right-click the entry, and select **Set Access Permissions**
3. Optionally, select **Show Inherited ACIs** to additionally display entries on higher levels of the directory.



18.8. ADDING AN ACI

This section describes how you can add an ACI.

18.8.1. Adding an ACI Using the Command Line

Use the **ldapmodify** utility to add an ACI. For example:

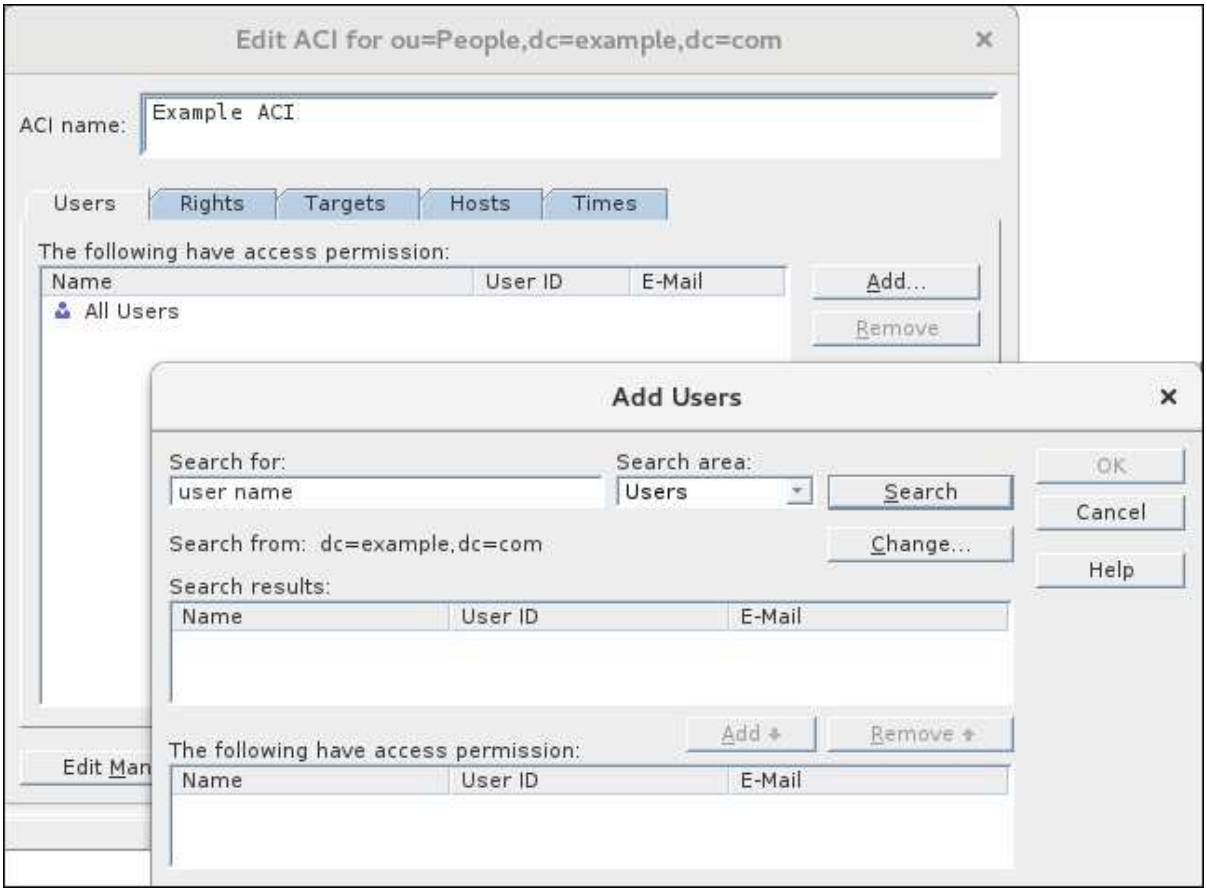
-

```
# 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 "Allow users updating their password";
 allow (write) userdn= "ldap:///self";)
```

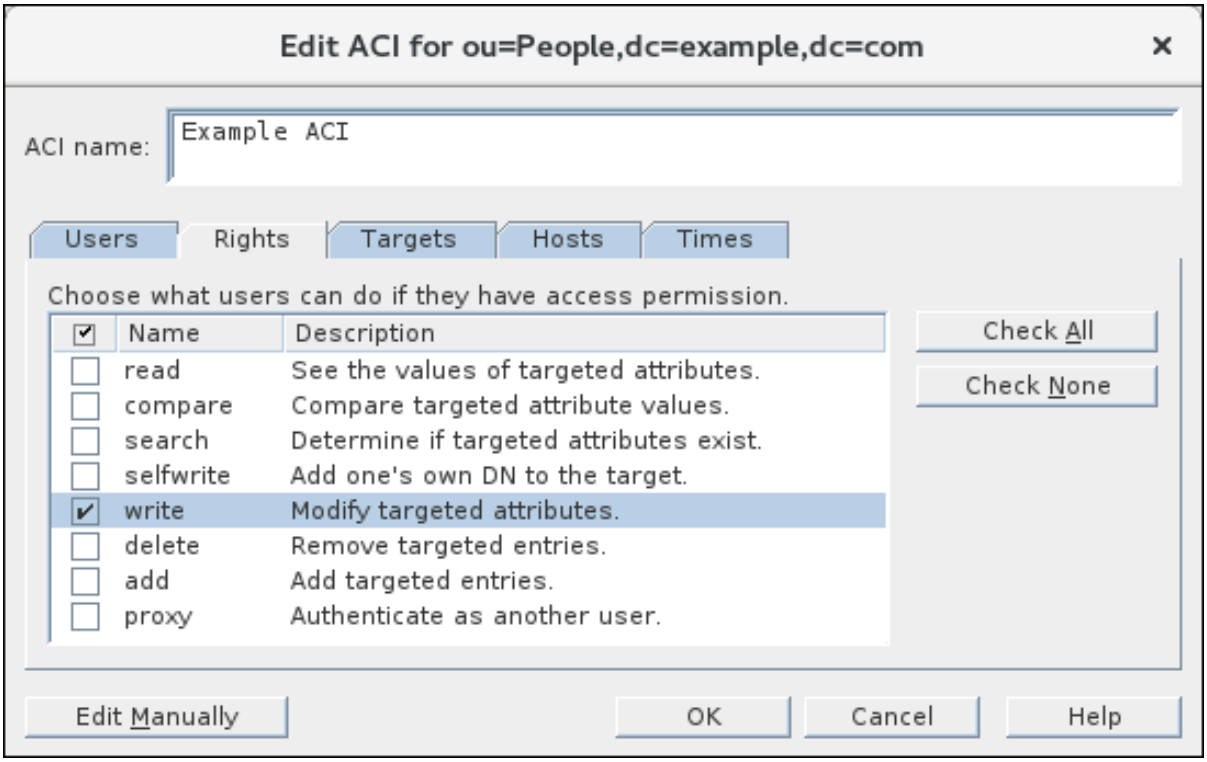
18.8.2. Adding an ACI Using the Console

To use the console to add an ACI:

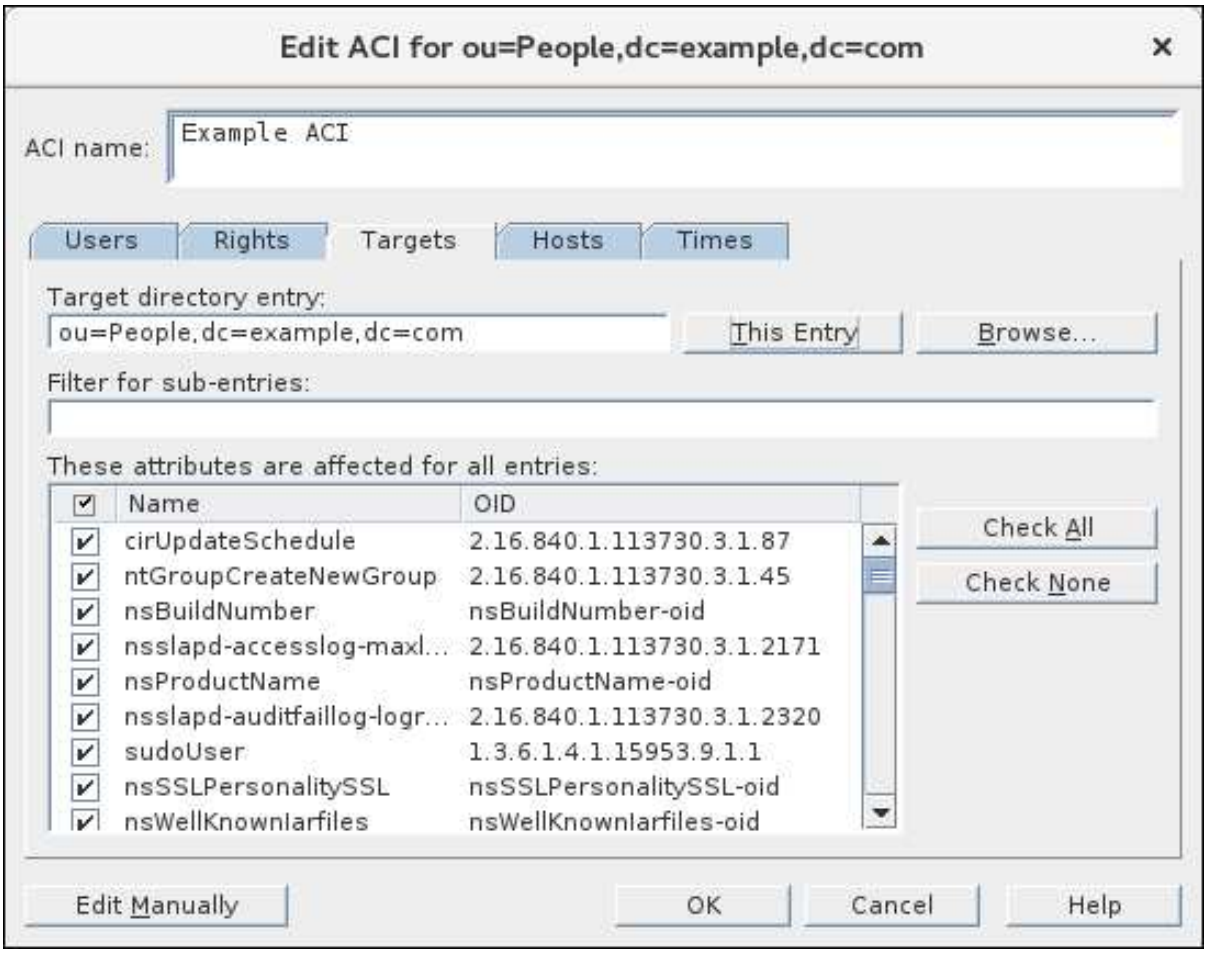
1. Open the Directory Server Console.
2. On the **Directory** tab, right-click the entry, and select **Set Access Permissions**
3. Enter the name of the ACI into the **ACI Name** field.
4. On the **Users** tab, optionally add users, groups, roles, administrators, or special rights to the list by clicking the **Add** button:
   a. Enter a string into the **Search for** field, select a search area, and click **Search**.
   b. Select the entry from the search results and click **Add**.
   c. Click **OK**.



5. On the **Rights** tab, select the permissions to set in this ACI.



6. On the **Targets** tab, select the target directory entry.



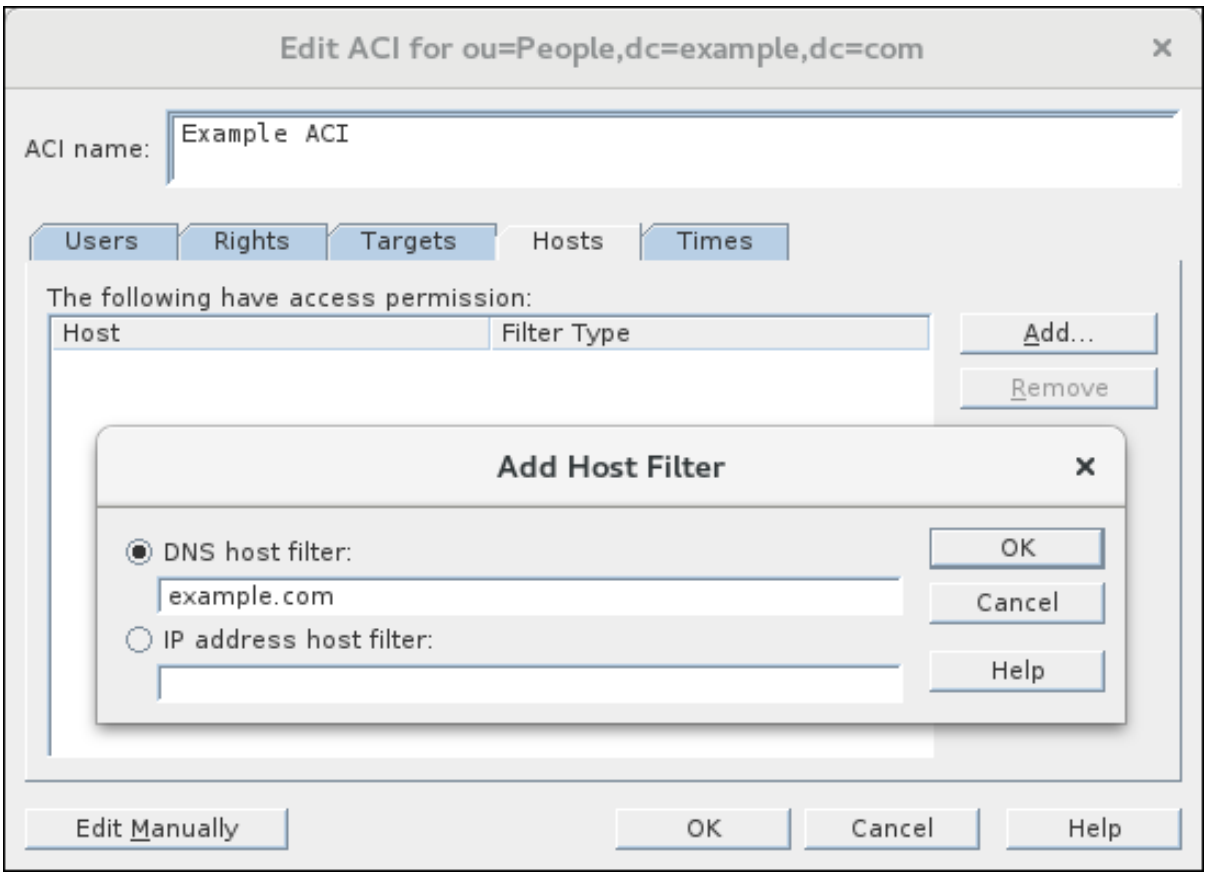
NOTE

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 ACIs to target every entry in the sub-tree under this node, enter a filter in the **Filter for Sub-entries** field. The filter applies to every entry below the target entry. For example, setting the filter to **ou=Sales** means that only entries with **ou=Sales** in their DN are returned.

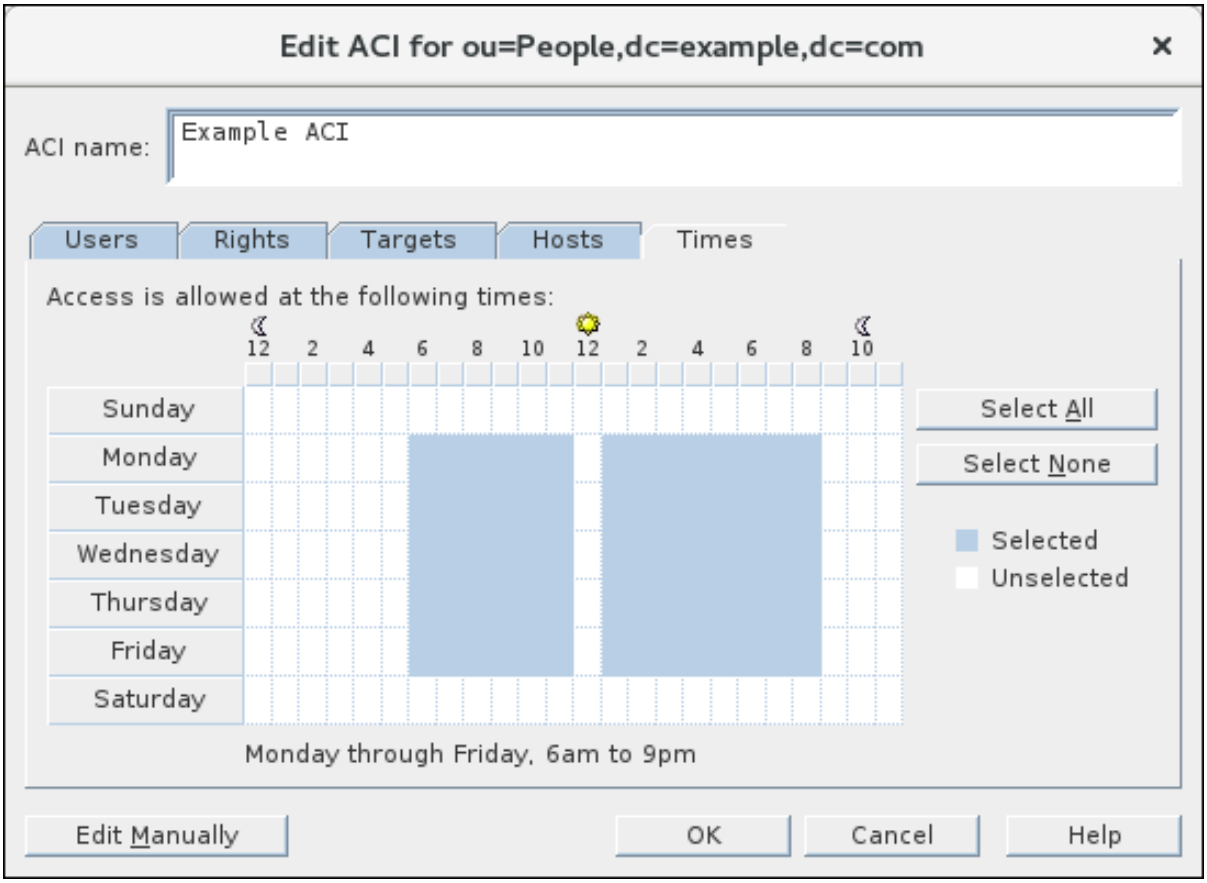
Additionally, you can restrict the scope of the ACI to certain attributes by selecting the attributes in the list.

7. On the **Hosts** tab, optionally add a DNS name or IP address.



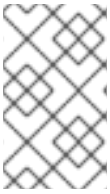
If you set a DNS name or IP address, the ACI applies only to LDAP operations from these hosts.

8. On the **Times** tab, optionally select at which times the ACI will be applied.



By default, access is allowed at all times. Change the access times by clicking and dragging the cursor over the table. Note that you can only select continuous time ranges.

9. Click **OK**.



NOTE

At any point of creating an ACI, click the **Edit Manually** button to display the LDIF statement corresponding to the wizard input. You can edit this statement in this window, however, the changes may not be visible in the graphical interface.

18.9. DELETING AN ACI

This section describes how you can delete an ACI from an entry.

18.9.1. Deleting an ACI Using the Command Line

To delete an ACI using the command line:

1. Display the ACIs set on the entry. See Section 18.7.1, “Displaying ACIs Using the Command Line”.

2. Delete the ACI:

   - If only one ***aci*** attribute is set on the entry or you want to remove all ACIs from the entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: delete
```

```
delete: aci
```

- If multiple ACIs exist on the entry and you want to delete a specific ACI, specify the exact ACI:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
delete: aci
aci: (targetattr="userPassword") (version 3.0; acl "Allow users
 updating their password"; allow (write) userdn= "ldap:///self";)
```

For further details about deleting attributes, see Section 3.1.4.3, “Deleting Attributes from an Entry” .

18.9.2. Removing an ACI Using the Console

To remove an ACI using the Console:

1. Open the Directory Server Console.
2. On the **Directory** tab, right-click the entry, and select **Set Access Permissions**
3. Select the ACI from the list and click **Remove**.
4. Click **OK**.

18.10. UPDATING AN ACI

This section describes how you can update an existing ACI.

18.10.1. Updating an ACI Using the Command Line

To update an ACI using the command line:

1. Delete the existing ACI. See Section 18.9.1, “Deleting an ACI Using the Command Line” .
2. Add a new ACI with the updated settings. See Section 18.8.1, “Adding an ACI Using the Command Line”.

18.10.2. Updating an ACI Using the Console

To update an ACI using the Console:

1. Open the Directory Server Console.
2. On the **Directory** tab, right-click the entry, and select **Set Access Permissions**
3. Select the ACI from the list and click **Edit**.
4. Update the ACI. The individual screens are described in the Section 18.8.2, “Adding an ACI Using the Console” section.
5. Click **OK**.

18.11. DEFINING TARGETS

Target rules in an ACI define to which entries Directory Server applies the ACI. If you do not set a target, the ACI applies to the entry containing the ***aci*** attribute and to entries below.

In an ACI, the following highlighted part is the target rule:

> (*target_rule*)(version 3.0; acl "*ACL_name*"; *permission_rule bind_rules*;)

For complex ACIs, Directory Server supports multiple target rules with different keywords:

> (*target_rule_1*)(*target_rule_2*)(...)(version 3.0; acl "*ACL_name*"; *permission_rule bind_rules*;)

If you specify multiple target rules, the order is not relevant. Note that you can use each of the following keywords only once in an ACI:

- **target**
- **targetattr**
- **targetattrfilters**
- **targetfilter**
- **target_from**
- **target_to**

Syntax

The general syntax of a target rule is:

> (*keyword comparison_operator* "*expression*")

- ***keyword***: Sets the type of the target. See Section 18.11.1, “Frequently Used Target Keywords” .
- ***comparisonoperator***: Valid values are **=** and **!=** and indicate whether or not the target is the object specified in the expression.

WARNING

For security reasons, Red Hat recommends not using the **!=** operator, because it allows the specified operation on all other entries or attributes. For example:

```
(targetattr != "userPassword");(version 3.0; acl "example"); allow (write)
... );
```

The previous example allows users to set, update, or delete any attribute except the ***userPassword*** attribute under the Distinguished Name (DN) you set the ACI. However, this enables users, for example, to add an additional ***aci*** attribute that allows write access to this attribute as well.

- ***expression***: Sets the target and must be surrounded by quotation marks. The expression itself depends on the keyword you use.

18.11.1. Frequently Used Target Keywords

Administrators frequently use the following target keywords:

- **target**: See Section 18.11.1.1, "Targeting a Directory Entry" .
- **targetattr**: See Section 18.11.1.2, "Targeting Attributes".
- **targetfilter**: See Section 18.11.1.3, "Targeting Entries and Attributes Using LDAP Filters" .
- **targattrfilters**: See Section 18.11.1.4, "Targeting Attribute Values Using LDAP Filters" .

18.11.1.1. Targeting a Directory Entry

To control access based on a DN and the entries below it, use the **target** keyword in the ACI. A target rule which uses the **target** keyword takes a DN as expression:

```
(target comparison_operator "ldap:///distinguished_name")
```

NOTE

You must set the ACI with the **target** keyword on the DN you are targeting or a higher-level DN of it. For example, if you target **ou=People,dc=example,dc=com**, you must either set the ACI on **ou=People,dc=example,dc=com** or **dc=example,dc=com**.

Example 18.1. Using the target Keyword

To enable users that are stored in the **ou=People,dc=example,dc=com** entry to search and display all attributes in their own entry:

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

```
dn: dc=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com") (version 3.0;
 acl "Allow users to read and search attributes of own entry"; allow (search, read)
 (userdn = "ldap:///self");)
```

Using Wildcards with the target Keyword

You can use the * wildcard character target multiple entries.

The following target rule example matches all entries in **ou=People,dc=example,dc=com** whose ***uid*** attribute is set to a value that starts with the letter **a**:

```
(target = "ldap:///uid=a*,ou=People,dc=example,dc=com")
```

Depending on the position of the wildcard, the rule not only applies to attribute values, but also to the full DN. Therefore, you can uses the wildcard as a substitute for portions of the DN.

Example 18.2. Targeting a Directory Entries Using Wildcards

The following rule targets all entries in the **dc=example,dc=com** tree with a matching ***uid*** attribute and not only entries which are stored in the **dc=example,dc=com** entry itself:

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

The previous target rule matches multiple entries, such as:

- **uid=*user_name*,dc=example,dc=com**
- **uid=*user_name*,ou=People,dc=example,dc=com**
- **uid=*user_name2*,dc=example,dc=com**

IMPORTANT

Directory Server does not support wildcards in the suffix part of a DN. For example, if your directory's suffix is **dc=example,dc=com**, you cannot use a target with a wildcard in this suffix, such as **(target = "ldap:///dc=*.com")**.

18.11.1.2. Targeting Attributes

To limit access in an ACI to certain attributes, use the **targetattr** keyword. For example, this keyword defines:

- In a read operation, what attributes will be returned to a client
- In a search operation, what attributes will be searched
- In a write operation, what attributes can be written to an object
- In an add operation, what attributes can be added when creating a new object

NOTE

In certain situations, you can use the **targetattr** keyword to secure ACIs by combining other target keywords with **targetattr**. See Section 18.11.3, "Advanced Usage of Target Rules" for examples.

To separate multiple attributes in a target rule that uses the **targetattr** keyword, use || command:

(targetattr *comparison_operator* "*attribute_1* || *attribute_2* || ...")

The attributes set in the expression must be defined in the schema.

NOTE

The attributes specified in the expression applies to the entry on which you create the ACI and to all entries below it.

Example 18.3. Using the targetattr Keyword

To enable users stored in **dc=example,dc=com** and all subentries to update the ***userPassword*** attribute in their own entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "userPassword") (version 3.0;
 acl "Allow users updating own userPassword";
 allow (write) (userdn = "ldap:///self");)
```

Using Wildcards with the targetattr Keyword

Using the * wildcard character, you can, for example, target all attributes:

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

WARNING

For security reasons, do not use wildcards with the **targetattr**, because it allows access to all attributes, including operational attributes. For example, if users can add or modify all attributes, users might create additional ACI and increase their own permissions.

18.11.1.3. Targeting Entries and Attributes Using LDAP Filters

To target a group of entries that match a certain criteria, use the **targetfilter** keyword with an LDAP filter:

```
(targetfilter comparison_operator "LDAP_filter")
```

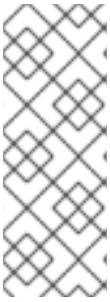
The filter expression is a standard LDAP search filter, as described in Chapter 14, *Finding Directory Entries*.

Example 18.4. Using the targetfilter Keyword

To grant permissions to members of the **cn=Human Resources,dc=example,dc.com** group to modify all entries having the ***department*** attribute set to **Engineering** or **Sales**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetfilter = "(|(department=Engineering)(department=Sales)")
 (version 3.0; acl "Allow HR updating engineering and sales entries";
 allow (write) (groupdn = "ldap:///cn=Human Resources,dc=example,dc.com");)
```

The **targetfilter** keyword targets whole entries. If you combine it with the **targetattr** keyword, the ACI applies only to a subset of attributes of the targeted entries. See Section 18.11.3.3, "Targeting Certain Attributes of Entries Matching a Filter".



NOTE

Using LDAP filters is useful when targeting entries and attributes that are spread across the directory. However, 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, verify that they target the correct entries and attributes by using the same filter, for example, in an **ldapsearch** operation.

Using Wildcards with the targetfilter Keyword

The **targetfilter** keyword supports wildcards similarly to standard LDAP filters. For example, to target all ***uid*** attributes whose value starts with **adm**:

```
(targetattr = "(uid=adm*) ...)
```

18.11.1.4. Targeting Attribute Values Using LDAP Filters

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

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

- For one operation with one attribute and filter combination:

  ```
  (targattrfilters="operation=attribute:filter")
  ```

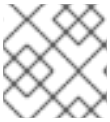
- For one operation with multiple attribute and filter combinations:

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

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

> (targattrfilters="*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.



NOTE

Value-based ACIs are only supported using the command line.

The following describes how filter must match:

- When creating an entry and a filter applies to an attribute in the new entry, then each instance of that attribute must match the filter.
- When deleting an entry and a filter applies to an attribute in the entry, then each instance of that attribute must also match the filter.
- When modifying an entry and the operation adds an attribute, then the **add** filter that applies to that attribute must match.
- If the operation deletes an attribute, then the **del** filter that applies to that attribute must match. If the individual values of an attribute already present in the entry are replaced, then both the **add** and **del** filters must match.

Example 18.5. Using the targattrfilters Keyword

To create an ACI that enables users to add any role to their own entry, except the **Admin** role, and to add the ***telephone*** attribute, as long as the value begins with the **123** prefix:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targattrfilters="add=nsroledn:(!(nsroledn=cn=Admin)) &&
 telephoneNumber:(telephoneNumber=123*)") (version 3.0;
 acl "Allow adding roles and telephone";
 allow (add) (userdn = "ldap:///self");)
```

18.11.2. Further Target Keywords

This section describes target keywords that are less-frequently used.

18.11.2.1. Targeting Source and Destination DNs

In certain situations, administrators want to allow users to move directory entries. Using the **target_from** and **target_to** keywords in an ACI, you can specify the source and destination of the operation, however, without enabling the user:

- To move entries from a different source as set in the ACI.
- To move entries to a different destination as set in the ACI.
- To delete existing entries from the source DN.
- To add new entries to the destination DN.

Example 18.6. Using the target_from and target_to Keywords

For example, to enable the **uid=user,dc=example,dc=com** account to move user accounts from the **cn=staging,dc=example,dc=com** entry to **cn=people,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target_from="ldap:///uid=*,cn=staging,dc=example,dc=com")
 (target_to="ldap:///cn=People,dc=example,dc=com")
 (version 3.0; acl "MODDN from"; allow (moddn))
 userdn="ldap:///uid=user,dc=example,dc=com";)
```

NOTE

ACIs apply only to the subtree where they are defined. In the previous example, the ACI is applies only to the **dc=example,dc=com** subtree.

If the **target_from** or **target_to** keyword is not set, the ACI matches any source or destination.

18.11.3. Advanced Usage of Target Rules

By combining multiple keywords, you can create complex target rules. This section provides examples of the advanced usage of target rules.

18.11.3.1. Delegating Permissions to Create and Maintain Groups

In certain situations, administrators want to delegate permissions to other accounts or groups. By combining target keywords, you can create secure ACIs that solve this request.

Example 18.7. Delegating Permissions to Create and Maintain Groups

To enable the **uid=*user*,ou=People,dc=example,dc=com"** account to create and update groups in the **ou=groups,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///cn=*,ou=Groups,dc=example,dc=com")
```

```
targetfilter="(&(objectClass=top)(objectClass=groupOfUniqueNames))")
(targetattr="cn || uniqueMember || objectClass")
(version 3.0; acl "example"; allow (read, search, write, add)
(userdn = "ldap:///uid=test,ou=People,dc=example,dc=com");)
```

For security reasons, the previous example adds certain limitations. The **uid=test,ou=People,dc=example,dc=com** user:

- Can create objects that must contain the **top** and **groupOfUniqueNames** object classes.
- Cannot add additional object classes, such as **account**. For example, this prevents if you use Directory Server accounts for local authentication, to create new users with an invalid user ID, such as **0** for the **root** user.

18.11.3.2. Targeting Both an Entry and Attributes

The **target** controls access based on a DN. However, if you use it in combination with a wildcard and the **targetattr** keyword, you can target both entries and attributes.

Example 18.8. Targeting Both an Entry and Attributes

To enable the **uid=user,ou=People,dc=example,dc.com** user to read and search members of groups in all organizational units in the **dc=example,dc=com** subtree:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=*,dc=example,dc=com")(targetattr="member" || "cn") (version 3.0;
 acl "Allow uid=user to search and read members of groups";
 allow (read, search) (userdn = "ldap:///uid=user,ou=People,dc=example,dc.com");)
```

18.11.3.3. Targeting Certain Attributes of Entries Matching a Filter

If you combine the **targetattr** and **targetfilter** keywords in two target rules, you can target certain attributes in entries that match a filter.

Example 18.9. Targeting Certain Attributes of Entries Matching a Filter

To allow members of the **cn=Engineering Admins,dc=example,dc=com** group to modify the ***jpegPhoto*** and ***manager*** attributes of all entries having the ***department*** attribute set to **Engineering**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "jpegPhoto|| manager")
 (targetfilter = "(department=Engineering)") (version 3.0;
 acl "Allow engineering admins updating jpegPhoto and manager of department members";
 allow (write) (groupdn = "ldap:///cn=Engineering Admins,dc=example,dc.com");)
```

18.11.3.4. Targeting a Single Directory Entry

To target a single directory entry, combine the **targetattr** and **targetfilter** keywords.

Example 18.10. Targeting a Single Directory Entry

To enable the **uid=user,ou=People,dc=example,dc=com** user to read and search the ***ou*** and ***cn*** attributes in the **ou=Engineering,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=Engineering,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "ou || cn")
 (targetfilter = "(ou=Engineering)") (version 3.0;
 acl "Allow uid=user to search and read engineering attributes";
 allow (read, search) (userdn = "ldap:///uid=user,ou=People,dc=example,dc.com");)
```

To enable the previous example to target only the **ou=Engineering,dc=example,dc=com** entry, sub-entries in **ou=Engineering,dc=example,dc=com** must not have the ***ou*** attribute set to **Engineering**.

IMPORTANT

These kind of ACIs can fail if the structure of your directory changes.

Alternatively, you can create a bind rule that matches the user input in the bind request with an attribute value that is stored in the targeted entry. See Section 18.13.2.1, “Defining Access Based on Value Matching”.

18.12. DEFINING PERMISSIONS

Permission rules define the rights that are associated with the ACI and whether access is allowed or denied.

In an ACI, the following highlighted part is the permission rule:

(*target_rule*) (version 3.0; acl "*ACL_name*"; *permission_rule bind_rules*;)

Syntax

The general syntax of a permission rule is:

permission (*rights*)

- ***permission***: Sets if the ACI allows or denies permission.
- ***rights***: Sets the rights which the ACI allows or denies. See Section 18.12.1, “User rights”.

Example 18.11. Defining Permissions

To enable users stored in the **ou=People,dc=example,dc=com** entry to search and display all attributes in their own entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=People,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ou=People,dc=example,dc=com") (version 3.0;
 acl "Allow users to read and search attributes of own entry"; allow (search, read)
 (userdn = "ldap:///self");)
```

18.12.1. User rights

The rights in a permission rule define what operations are granted or denied. In an ACI, you can set one or multiple of the following rights:

Table 18.1. User Rights

Right	Description
read	Sets whether users can read directory data. This permission applies only to search operations in LDAP.
write	Sets whether users can modify an entry by adding, modifying, or deleting attributes. This permission applies to the **modify** and **modrdn** operations in LDAP.
add	Sets whether users can create an entry. This permission applies only to the **add** operation in LDAP.
delete	Sets whether users can delete an entry. This permission applies only to the **delete** operation in LDAP.
search	Sets whether users can search for directory data. To view data returned as part of a search result, assign **search** and **read** rights. This permission applies only to search operations in LDAP.
compare	Sets 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 in LDAP.
selfwrite	Sets whether users can add or delete their own DN from a group. This right is used only for group management.

Right	Description
proxy	Sets whether the specified DN can access the target with the rights of another entry. The **proxy** right is granted within the scope of the ACL, and the user or group who as the right granted can run commands as any Directory Server user. You cannot restrict the proxy rights to certain users. For security reasons, set ACIs that use the **proxy** right at the most targeted level of the directory.
all	Sets all of the rights, except **proxy**.

18.12.2. Rights Required for LDAP Operations

This section describes the rights you must 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 that you want to add.
  - Grant **write** permission on the value of each attribute in the entry. This right is granted by default but can be restricted using the **targattrfilters** keyword.
- Deleting an entry:
  - Grant **delete** permission on the entry that you want to delete.
  - Grant **write** permission on the value of each attribute in the entry. This right is granted by default but can be restricted using the **targattrfilters** 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 can be restricted using the **targattrfilters** keyword.
- Modifying the RDN of an entry:
  - Grant **write** permission on the entry.
  - Grant **write** permission on the attribute type that is used in the new RDN.
  - Grant **write** permission on the attribute type that is 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 that is used in the new RDN. This right is granted by default but can be restricted using the **targattrfilters** 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.

18.12.3. Access Control and the modrdn Operation

To explicitly deny **modrdn** operations using ACIs, target the relevant entries but omit the **targetattr** keyword. For example, to add an ACI that defines the **cn=example,ou=Groups,dc=example,dc=com** group, cannot rename entries in **ou=people,dc=example,dc=com** which contain the ***cn*** attribute:

```
ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (target="ldap:///cn=*,ou=people,dc=example,dc=com")
 (version 3.0; acl "Deny modrdn rights to the example group";
 deny(write) groupdn="ldap:///cn=example,ou=groups,dc=example,dc=com";)
```

18.13. DEFINING BIND RULES

The bind rules in an ACI define the required bind parameters that must meet so that Directory Server applies the ACI. For example, you can set bind rules based on:

- DNs
- Group memberships or assigned roles
- Locations from which an entry must bind
- Types of authentication that must be in use during the bind
- Times or days on which the bind occurs

In an ACI, the following highlighted part is the bind rule:

```
(target_rule) (version 3.0; acl "ACL_name"; permission_rule bind_rules;)
```

Syntax
The general syntax of a bind rule is:

```
keyword comparison_operator "expression"
```

- ***keyword***: Sets the type of the bind operation. See Section 18.13.1, “Frequently Used Bind Rules” .
- ***comparison_operator***: Valid values are **=** and **!=** and indicate whether or not the target is the object specified in the expression. If a keyword supports additional comparison operators, it is mentioned in the corresponding section.
- ***expression***: Sets the expression and must be surrounded by quotation marks. The expression itself depends on the keyword you use.

18.13.1. Frequently Used Bind Rules

Administrators frequently use the following bind keywords:

- **userdn**: See Section 18.13.1.1, “Defining User-based Access” .
- **groupdn**: See Section 18.13.1.2, “Defining Group-based Access” .

Additionally, bind rules are frequently combined using Boolean operators. For details, see Section 18.13.3, “Combining Bind Rules Using Boolean Operators” .

18.13.1.1. Defining User-based Access

The **userdn** keyword enables you to grant or deny access based on one or multiple DNs and uses the following syntax:

```
userdn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."
```

Set the DN in the expression to:

- A DN: See Section 18.13.1.1.1, “Using a DN with the **userdn** Keyword”.
- An LDAP filter: See Section 18.13.1.1.2, “Using the **userdn** Keyword with an LDAP filter” .
- The **anyone** alias: See Section 18.13.1.1.3, “Granting Anonymous Access” .
- The **all** alias: See Section 18.13.1.1.4, “Granting Access to Authenticated Users” .
- The **self** alias: See Section 18.13.1.1.5, “Enabling Users to Access Their Own Entries” .
- The **parent** alias: See Section 18.13.1.1.6, “Setting Access for Child Entries of a User” .

NOTE

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

18.13.1.1.1. Using a DN with the**userdn** Keyword

Set the **userdn** keyword to a DN to apply the ACI only to the matching entry. To match multiple entries, use the * wildcard in the DN.

Using the **userdn** keyword with a DN must match the following syntax:

```
userdn comparison_operator ldap:///distinguished_name
```

Example 18.12. Using a DN with the userdn Keyword

To enable the **uid=admin,ou=People,dc=example,dc=com** user to read the ***manager*** attribute of all other users in the **ou=People,dc=example,dc=com** entry:

```
# 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="manager") (version 3.0; acl "Allow uid=admin reading manager attribute";
 allow (search, read) userdn = "ldap:///uid=admin,ou=People,dc=example,dc=com";)
```

18.13.1.1.2. Using the **userdn** Keyword with an LDAP filter

If you want to dynamically allow or deny permissions to users, use the **userdn** keyword with an LDAP filter:

```
userdn comparison_operator "ldap:///distinguished_name??scope?(filter)"
```

NOTE

The LDAP filter supports the * wildcard.

Example 18.13. Using the userdn Keyword with an LDAP filter

To enable users who have the ***department*** attribute set to **Human Resources** to update the ***homePostalAddress*** attribute of users in the **ou=People,dc=example,dc=com** entry:

```
# 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="homePostalAddress") (version 3.0;
 acl "Allow HR setting homePostalAddress"; allow (write)
 userdn = "ldap:///ou=People,dc=example,dc=com??sub?(department=Human Resources)";)
```

18.13.1.1.3. Granting Anonymous Access

In certain situations, administrators want to configure anonymous access to data in the directory. Anonymous access means that it is possible to bind to the directory by providing:

- No bind DN and password
- A valid bind DN and password

To configure anonymous access, use the **ldap:///anyone** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap:///anyone"
```

Example 18.14. Granting Anonymous Access

To enable anyone without authentication to read and search the ***sn***, ***givenName***, and ***telephoneNumber*** attributes in the **ou=People,dc=example,dc=com** entry:

```
# 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="sn" || targetattr="givenName" || targetattr = "telephoneNumber")
 (version 3.0; acl "Anonymous read, search for names and phone numbers";
 allow (read, search) userdn = "ldap:///anyone")
```

18.13.1.1.4. Granting Access to Authenticated Users

In certain situations, administrators want to grant permission to any user who is able to successfully bind to Directory Server, except anonymous binds. To configure this feature, use the **ldap:///all** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap:///all"
```

Example 18.15. Granting Access to Authenticated Users

To enable authenticated users to add and remove themselves as a member from the **ou=example,ou=groups,dc=example,dc=com** group:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=example,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="member") (version 3.0;
 acl "Allow users to add/remove themselves from example group";
 allow (selfwrite) userdn = "ldap:///all")
```

18.13.1.1.5. Enabling Users to Access Their Own Entries

To set ACIs which allow or deny access to users to their own entry, use the **ldap:///self** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap:///self"
```

Example 18.16. Enabling Users to Access Their Own Entries

To enable users in the **ou=People,dc=example,dc=com** entry to update their own ***userPassword*** attribute:

```
# 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 "Allow users updating their password";
 allow (write) userdn = "ldap:///self")
```

18.13.1.1.6. Setting Access for Child Entries of a User

To specify that users are granted or denied access to an entry only if their bind DN is the parent of the targeted entry, use the **self:///parent** expression with the **userdn** keyword in a bind rule:

```
userdn comparison_operator "ldap:///parent"
```

Example 18.17. Setting Access for Child Entries of a User

To enable the **cn=*user*,ou=People,dc=example,dc=com** user to update the ***manager*** attribute of its own sub-entries, such as **cn=example,cn=*user*,ou=People,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=user,ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0;
 acl "Allow cn=user to update manager attributes";
 allow (write) userdn = "ldap:///parent")
```

18.13.1.2. Defining Group-based Access

Group-based ACIs enable you to manage access by adding or removing users to or from a group. To configure an ACI that is based on a group membership, use the **groupdn** keyword. If the user is a member of one or multiple of the specified groups, the ACI matches.

When using the **groupdn** keyword, Directory Server verifies the group membership based on the following attributes:

- ***member***
- ***uniqueMember***
- ***memberURL***
- ***memberCertificateDescription***

Bind rules with the **groupdn** keyword use the following syntax:

```
groupdn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."
```

Set the DN in the expression to:

- A DN. See Section 18.13.1.2.1, “Using a DN with the **groupdn** Keyword”.
- An LDAP filter. See Section 18.13.1.2.2, “Using The **groupdn** Keyword with an LDAP Filter” .

If you set multiple DNs in one bind rule, Directory Server applies the ACI if the authenticated user is a member of one of these groups. To set the user as a member of multiple groups, use multiple **groupdn** keywords and combine them using the Boolean **and** operator. For details, see Section 18.13.3, “Combining Bind Rules Using Boolean Operators”.

NOTE

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

18.13.1.2.1. Using a DN with the groupdn Keyword

To apply an ACI to members of a group, set the **groupdn** keyword to the group's DN.

The **groupdn** keyword set to a DN uses the following syntax:

-

groupdn *comparison_operator* ldap:///*distinguished_name*

Example 18.18. Using a DN with the groupdn Keyword

To enable members of the **cn=example,ou=Groups,dc=example,dc=com** group to search and read the ***manager*** attribute of entries in **ou=People,dc=example,dc=com**:

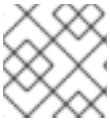
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=People,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="manager") (version 3.0;
 acl "Allow example group to read manager attribute";
 allow (search, read) groupdn = "ldap:///cn=example,ou=Groups,dc=example,dc=com";)
```

18.13.1.2.2. Using The groupdn Keyword with an LDAP Filter

Using an LDAP filter with the **groupdn** keyword, you can define that the authenticated user must be a member of at least one of the groups that the filter search returns, to match the ACI.

The **groupdn** keyword with an LDAP filter uses the following syntax:

groupdn *comparison_operator* "ldap:///*distinguished_name*??*scope*?(*filter*)"



NOTE

The LDAP filter supports the * wildcard.

Example 18.19. Using The groupdn Keyword with an LDAP Filter

To enable members of groups in **dc=example,dc=com** and subtrees, which have the ***manager*** attribute set to **example**, update the ***homePostalAddress*** of entries in **ou=People,dc=example,dc=com**:

```
# 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="homePostalAddress") (version 3.0;
 acl "Allow manager=example setting homePostalAddress"; allow (write)
 userdn = "ldap:///dc=example,dc=com??sub?(manager=example)";)
```

18.13.2. Further Bind Rules

This section describes bind rules that are less-frequently used.

18.13.2.1. Defining Access Based on Value Matching

Use the **userattr** keyword in a bind rule to specify which attribute must match between the entry used to bind to the directory and the targeted entry.

The **userattr** keyword uses the following syntax:

```
userattr comparison_operator "attribute_name#bind_type_or_attribute_value
```

For further details, see:

- Section 18.13.2.1.1, "Using the **USERDN** Bind Type"
- Section 18.13.2.1.2, "Using the **GROUPDN** Bind Type"
- Section 18.13.2.1.3, "Using the **ROLEDN** Bind Type"
- Section 18.13.2.1.4, "Using the **SELFDN** Bind Type"
- Section 18.13.2.1.5, "Using the **LDAPURL** Bind Type"
- Section 18.13.2.1.6, "Matching an Attribute's Value of the Bind DN and Target DN"

IMPORTANT

By default, Directory Server evaluates access rights on the entry they are created. However, to prevent user objects on the same level, Directory Server does not grant **add** permissions to the entry where you set the ACI, when using the **userattr** keyword. To configure this behavior, use the **userattr** keyword in conjunction with the **parent** keyword and grant the permission additionally on level **0**.

For details about inheritance, see Section 18.13.2.1.7, "Using the **userattr** Keyword with Inheritance".

18.13.2.1.1. Using the USERDN Bind Type

To apply an ACI when the binding user DN matches the DN stored in an attribute, use the **USERDN** bind type.

The **userattr** keyword with the **USERDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#USERDN"
```

Example 18.20. Using the USERDN Bind Type

To grant a manager all permissions to the ***telephoneNumber*** attribute of its own associates:

```
# 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 = "telephoneNumber")
 (version 3.0; acl "Manager: telephoneNumber";
 allow (all) userattr = "manager#USERDN";)
```

The previous ACI is evaluated to be true if the DN of the user who performs the operation on an entry in **ou=People,dc=example,dc=com**, matches the DN stored in the ***manager*** attribute of this entry.

18.13.2.1.2. Using the GROUPDN Bind Type

To apply an ACI when the binding user DN is a member of a group set in an attribute, use the **GROUPDN** bind type.

The **userattr** keyword with the **GROUPDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#GROUPDN"
```

Example 18.21. Using the GROUPDN Bind Type

To grant users the permission to delete a group entry which they own under the **ou=Social Committee,ou=Groups,dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=Social Committee,ou=Groups,dc=example,dc=com
changetype: modify
add: aci
aci: (target="ou=Social Committee,ou=Groups,dc=example,dc=com)
 (targattrfilters="del=objectClass:(objectClass=groupOfNames)")
 (version 3.0; acl "Delete Group";
 allow (delete) userattr = "owner#GROUPDN";)
```

The previous ACI is evaluated to be true if the DN of the user who performs the operation is a member of the group specified in the ***owner*** attribute.

The specified group 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, use the following expression for better performance:

```
userattr comparison_operator "ldap:///distinguished_name?attribute_name#GROUPDN"
```

18.13.2.1.3. Using the ROLEDN Bind Type

To apply an ACI when the binding user belongs to a role specified in an attribute, use the **ROLEDN** bind type.

The **userattr** keyword with the **ROLEDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#ROLEDN"
```

Example 18.22. Using the ROLEDN Bind Type

To enable users with the **cn=Administrators,dc=example,dc=com** role to search and read the ***manager*** attribute of entries in **ou=People,dc=example,dc=com**:

```
# 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: (version 3.0; acl "Allow example role owners to read manager attribute";
 allow (search, read) roledn="ldap:///cn=Administrators,dc=example,dc=com";)
```

The specified 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, use the following expression for better performance:

```
userattr comparison_operator "ldap:///distinguished_name?attribute_name#ROLEDN"
```

18.13.2.1.4. Using the SELFDN Bind Type

The **SELFDN** bind type enables you to grant permissions, when the bound user's DN is set in a single-value attribute of the entry.

The **userattr** keyword with the **SELFDN** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#SELFDN"
```

Example 18.23. Using the SELFDN Bind Type

To enable a user to add **ipatokenuniqueid=*,cn=otp,dc=example,dc=com** entries that have the bind user's DN set in the ***ipatokenOwner*** attribute:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: ou=otp,dc=example,dc=com
changetype: modify
add: aci
aci: (target = "ldap:///ipatokenuniqueid=*,cn=otp,dc=example,dc=com")
 (targetfilter = "(objectClass=ipaToken)")(version 3.0;
 acl "token-add-delete"; allow (add) userattr = "ipatokenOwner#SELFDN";)
```

18.13.2.1.5. Using the LDAPURL Bind Type

To apply an ACL when the bind DN matches the filter specified in an attribute of the targeted entry, use the **LDAPURL** bind type.

The **userattr** keyword with the **LDAPURL** bind type requires the following syntax:

```
userattr comparison_operator "attribute_name#LDAPURL"
```

Example 18.24. Using the LDAPURL Bind Type

To grant read and search permissions to user objects which contain the ***aciurl*** attribute set to **ldap:///ou=People,dc=example,dc=com??one?(uid=*user**)**

```
# 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 = "*")
 (version 3.0; acl "Allow read,search "; allow (read,search)
 (userattr = "aciurl#LDAPURL);)
```

18.13.2.1.6. Matching an Attribute's Value of the Bind DN and Target DN

To apply an ACL when both the bind DN entry and the targeted entry contain an attribute set to the same value, use the following syntax:

```
userattr comparison_operator "attribute_name#value"
```

Example 18.25. Matching an Attribute's Value of the Bind DN and Target DN

To grant read and search permissions to both user performing operation and user in the tree with the ***l*** attribute set to **office_1**:

```
# 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 "Users in the same location";
 allow (read,search) userattr = "l#office_1";)
```

18.13.2.1.7. 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 certain situations, administrators 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 using the **userattr** keyword with the **parent** keyword, the syntax is as follows:

```
userattr comparison_operator
"parent[inheritance_level].attribute_name#bind_type_or_attribute_value
```

- *inheritance_level*: Comma-separated list that indicates how many levels below the target inherit the ACI. You can include five levels (**0**, **1**, **2**, **3**, **4**) below the targeted entry. Zero (**0**) indicates the targeted entry.
- *attribute_name*: The attribute targeted by the **userattr** or **groupattr** keyword.
- *bind_type_or_attribute_value*: Sets the attribute value or a bind type, such as **USERDN**.

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.

Example 18.26. Using the userattr Keyword with Inheritance

To enable a user to read and search the **cn=Profiles,dc=example,dc=com** entry where the user's DN is set in the ***owner*** attribute, as well as the first level of child entries which includes **cn=mail,cn=Profiles,dc=example,dc=com** and **cn=news,cn=Profiles,dc=example,dc=com**:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=Profiles,dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr="*") (version 3.0; acl "Profile access",
 allow (read,search) userattr="parent[0,1].owner#USERDN" ;)
```

18.13.2.2. Defining Access from Specific IP Addresses or Ranges

The **ip** keyword in a bind rule enables you to grant or deny access from a specific IP address or a range of IP addresses.

Bind rules with the **ip** keyword use the following syntax:

```
ip comparison_operator "IP_address_or_range"
```

Example 18.27. Using IPv4 Address Ranges in Bind Rules

To deny access from the **192.0.2.2/24** network to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny 192.0.2.2/24"; deny (all)
 (userdn = "ldap:///anyone") and (ip != "192.0.2.");)
```

Example 18.28. Using IPv6 Address Ranges in Bind Rules

To deny access from the **2001:db8::/64** network to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny 2001:db8::/64"; deny (all)
 (userdn = "ldap:///anyone") and (ip != "2001:db8::");)
```

18.13.2.3. Defining Access from a Specific Host or Domain

The **dns** keyword in a bind rule enables you to grant or deny access from a specific host or domain.

WARNING

If Directory Server cannot resolve a connecting IP address to its Fully Qualified Domain Name (FQDN) using DNS, the server does not apply ACIs with the **dns** bind rule for this client.

If client IP addresses are not resolvable using DNS, use the **ip** keyword and IP addresses instead. See Section 18.13.2.2, "Defining Access from Specific IP Addresses or Ranges".

Bind rules with the **dns** keyword use the following syntax:

```
dns comparison_operator "host_name_or_domain_name"
```

Example 18.29. Defining Access from a Specific Host

To deny access from the **client.example.com** host to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny client.example.com"; deny (all)
 (userdn = "ldap:///anyone") and (dns != "client.example.com");)
```

Example 18.30. Defining Access from a Specific Domain

To deny access from all hosts within the **example.com** domain to the **dc=example,dc=com** entry:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "*") (version 3.0;acl "Deny example.com"; deny (all)
 (userdn = "ldap:///anyone") and (dns != "*.example.com");)
```

18.13.2.4. Requiring a Certain Level of Security in Connections

The security of a connection is determined by its Security Strength Factor (SSF), which sets the minimum key strength required to process operations. Using the **ssf** keyword in a bind rule, you can set that a connection must use a certain level of security. This enables you to force operations, for example password changes, to be performed over an encrypted connection.

The value for the SSF for any operation is the higher of the values between a 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.

Bind rules with the **ssf** keyword use the following syntax:

```
ssf comparison_operator key_strength
```

You can use the following comparison operators:

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

If the ***key_strength*** parameter is set to **0**, no secure operation is required for the LDAP operation.

Example 18.31. Requiring a Certain Level of Security in Connections

To configure that users in the **dc=example,dc=com** entry can only update their ***userPassword*** attribute when the SSF is **128** or higher:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: dc=example,dc=com
changetype: modify
add: aci
aci: (targetattr = "userPassword") (version 3.0;
 acl "Allow users updating own userPassword";
 allow (write) (userdn = "ldap:///self") (ssf >= "128");)
```

18.13.2.5. Defining Access at a Specific Day of the Week

The **dayofweek** keyword in a bind rule enables you to grant or deny access based on the day of the week.

NOTE

Directory Server uses the time on the server to evaluate the ACI; not the time on the client.

Bind rules with the **dayofweek** keyword use the following syntax:

```
dayofweek comparison_operator "comma-separated_list_of_days"
```

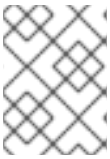
Example 18.32. Granting Access on Specific Days of the Week

To deny access for the **uid=*user*,ou=People,dc=example,dc=com** user entry to bind to the server on Saturdays and Sundays:

```
# 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: (version 3.0; acl "Deny access on Saturdays and Sundays";
 deny (all)
 (userdn = "ldap:///uid=user,ou=People,dc=example,dc=com") and
 (dayofweek = "Sun,Sat");)
```

18.13.2.6. Defining Access at a Specific Time of Day

The **timeofday** keyword in a bind rule enables you to grant or deny access based on the time of day.



NOTE

Directory Server uses the time on the server to evaluate the ACI; not the time on the client.

Bind rules with the **timeofday** keyword use the following syntax:

```
timeofday comparison_operator "time"
```

You can use the following comparison operators:

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

IMPORTANT

The **timeofday** keyword requires that you specify the time in 24-hour format.

Example 18.33. Defining Access at a Specific Time of a Day

To deny access for the **uid=*user*,ou=People,dc=example,dc=com** user entry to bind to the server between 6pm and 0am:

```
# 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: (version 3.0; acl "Deny access between 6pm and 0am";
 deny (all)
 (userdn = "ldap:///uid=user,ou=People,dc=example,dc=com") and
 (timeofday >= "1800" and timeofday < "2400");)
```

18.13.2.7. Defining Access Based on the Authentication Method

The **authmethod** keyword in a bind rule sets what authentication method a client must use when connecting to the server, to apply the ACI.

Bind rules with the **auth** keyword use the following syntax:

```
authmethod comparison_operator "authentication_method"
```

You can set the following authentication methods:

- **none**: Authentication is not required and represents anonymous access. This is the default.
- **simple**: The client must provide a user name and password to bind to the directory.
- **SSL**: The client must bind to the directory using a TLS certificate either in a database, smart card, or other device. For details about certificate-based authentication, see Section 9.8, “Using Certificate-based Client Authentication”.
- **SASL**: The client must bind to the directory over a Simple Authentication and Security Layer (SASL) connection. When you use this authentication method in a bind rule, additionally specify the SASL mechanism, such as **EXTERNAL**.

Example 18.34. Enabling Access Only for Connections Using the EXTERNAL SASL Authentication Method

To deny access to the server if the connection does not use a certificate-based authentication method or SASL:

```
# 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: (version 3.0; acl "Deny all access without certificate"; deny (all)
 (authmethod = "none" or authmethod = "simple");)
```

18.13.2.8. Defining Access Based on Roles

The **roledn** keyword in a bind rule enables you to grant or deny access to users having one or multiple role set.

NOTE

Red Hat recommends to use groups instead of roles. For further details about roles and limitations, see Section 8.2.1, “About Roles”.

Bind rules with the **roledn** keyword use the following syntax:

```
userdn comparison_operator "ldap:///distinguished_name || ldap:///distinguished_name || ..."
```

NOTE

If a DN contains a comma, escape the comma with a backslash.

Example 18.35. Defining Access Based on Roles

To enable users that have the **cn=Human Resources,ou=People,dc=example,dc=com** role set in the ***nsRole*** attribute to search and read the ***manager*** attribute of entries in **ou=People,dc=example,dc=com**:

```
# 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="manager") (version 3.0;
 acl "Allow manager role to update manager attribute";
 allow (search, read) roledn = "ldap:///cn=Human Resources,ou=People,dc=example,dc=com";)
```

18.13.3. Combining Bind Rules Using Boolean Operators

When creating complex bind rules, the **AND**, **OR**, and **NOT** Boolean operators enable you to combine multiple keywords.

Bind rules combined with Boolean operators have the following syntax:

```
bind_rule_1 boolean_operator bind_rule_2...
```

Example 18.36. Combining Bind Rules Using Boolean Operators

To configure that users which are member of both the **cn=Administrators,ou=Groups,dc=example,com** and **cn=Operators,ou=Groups,dc=example,com** group can read, search, add, update, and delete entries in **ou=People,dc=example,dc=com**:

```
# 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: (target="ldap:///ou=People,dc=example,dc=com") (version 3.0;
 acl "Allow members of administrators and operators group to manage users";
```

```
allow (read, search, add, write, delete)
groupdn = "ldap:///cn=Administrators,ou=Groups,dc=example,com" AND
groupdn = "ldap:///cn=Operators,ou=Groups,dc=example,com";)
```

How Directory Server Evaluates Boolean Operators

Directory Server evaluates Boolean operators by using the following rules:

- All expressions from left to right.

  In the following example, ***bind_rule_1*** is evaluated first:

  ```
  (bind_rule_1) OR (bind_rule_2)
  ```

- From innermost to outermost parenthetical expressions first.

  In the following example, ***bind_rule_2*** is evaluated first and ***bind_rule_3*** second:

  ```
  (bind_rule_1) OR ((bind_rule_2) AND (bind_rule_3))
  ```

- **NOT** before **AND** or **OR** operators.

  In the following example, ***bind_rule_2*** is evaluated first:

  ```
  (bind_rule_1) AND NOT (bind_rule_2)
  ```

The **AND** and **OR** operators have no order of precedence.

18.14. 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*.

18.14.1. 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 18.2, "Entry Rights" and Table 18.3, "Attribute Rights" show the access rights to entries and attributes, respectively, that are returned by a get effective rights search.

Table 18.2. Entry Rights

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

Table 18.3. 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.

18.14.2. 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 -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 18.14.3.2, "Examples of Get Effective Rights Searches for Non-Existent Attributes" and Section 18.14.3.3, "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.

18.14.3. 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.

18.14.3.1. 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 18.37. 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 -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 18.38, "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 18.38. 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 -
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 18.39, "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 18.39. 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 -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 -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 18.40, "Checking the Rights Someone Else Has to a Personal Entry" , Ted Morris checks what rights Dave Miller has on Ted Morris's entry.

Example 18.40. 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 -
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.

18.14.3.2. 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 18.41. Returning Effective Rights for Non-Existent Attributes

```
# ldapsearch -D "cn=Directory Manager" -W -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.

18.14.3.3. 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 18.14.2, "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 18.42. Get Effective Rights Results for Specific Attributes

```
# ldapsearch -D "cn=Directory Manager" -W -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
```

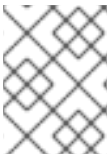
It is possible to specify a non-existent attribute in the *attributeList*, as with the ***initials*** attribute in Example 18.42, "Get Effective Rights Results for Specific Attributes" , to see the rights which are available, similar to using an asterisk to list all attributes.

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 18.43. Get Effective Rights Results for an Attribute within an Object Class

```
# ldapsearch -D "cn=Directory Manager" -W -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 18.44. Get Effective Rights Results for All Attributes for an Object Class

```
# ldapsearch -D "cn=Directory Manager" -W -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
```

18.14.3.4. 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 -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
```

18.14.3.5. 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 18.45. Get Effective Rights Results for Operational Attributes

```
# ldapsearch -D "cn=Directory Manager" -W -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
```

18.14.3.6. 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 18.46. Get Effective Rights Results with No ACL Set (Directory Manager)

```
# ldapsearch -D "cn=Directory Manager" -W -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 18.47. Get Effective Rights Results with No ACL Set (Regular User)

```
# ldapsearch -D "uid=scarter,ou=people,dc=example,dc=com" -W -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"
```

18.14.4. 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.

18.14.5. 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 18.4, "Returned Result Codes" summarizes the error codes and the potential configuration information they can relay.

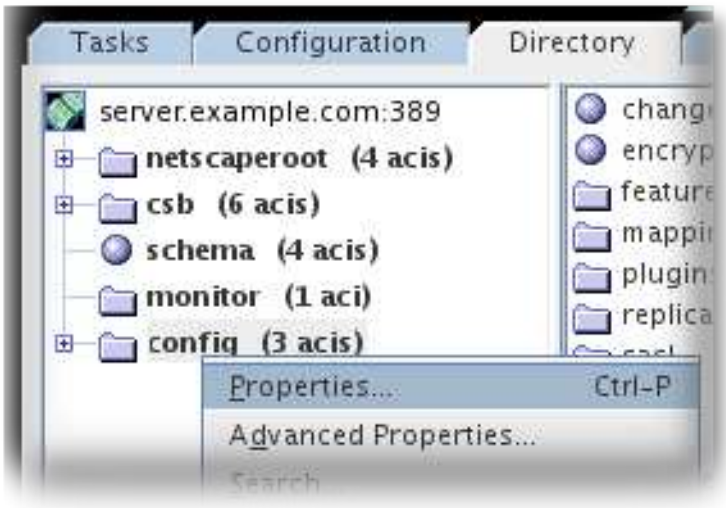
Table 18.4. Returned Result Codes

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.

18.15. 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.

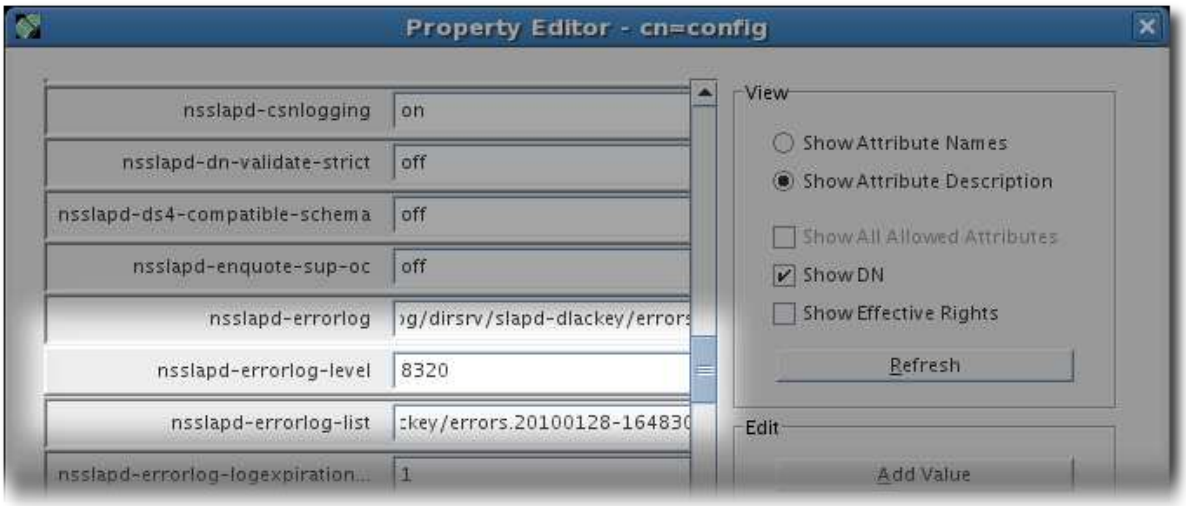


This displays the **Property Editor** for the **cn=config** entry.

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 *Red Hat Directory Server Configuration, Command, and File Reference*.



4. Click **OK** to dismiss the **Property Editor**.

18.16. 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.

18.16.1. Macro ACI Example

Figure 18.1, "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.

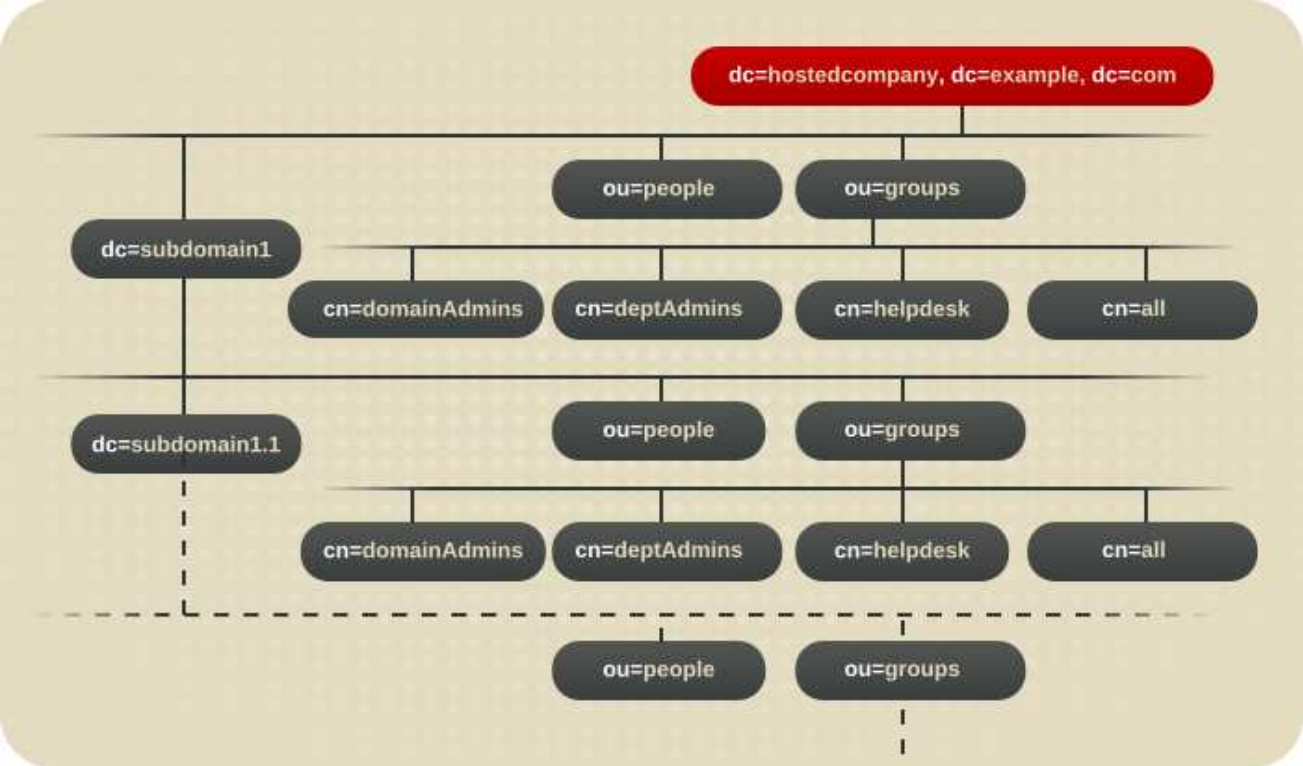


Figure 18.1. Example Directory Tree for Macro ACIs

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.

18.16.2. 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 18.5, “Macros in ACI Keywords” shows in what parts of the ACI you can use DN macros:

Table 18.5. Macros in ACI Keywords

Macro	ACI Keyword
($dn)	target, targetfilter, userdn, roledn, groupdn, userattr

Macro	ACI Keyword
[$dn]	targetfilter, userdn, roledn, groupdn, userattr
($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.

18.16.2.1. 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.

18.16.2.2. 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.

18.16.2.3. 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"
```

18.17. 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.

18.17.1. 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 18.11, “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 18.13.2.5, "Defining Access at a Specific Day of the Week" and Section 18.13.2.6, "Defining Access at a Specific Time of Day".
- IP address rules, where only specified IP addresses, domains, or subnets are explicitly allowed or denied. This is analogous to Section 18.13.2.2, "Defining Access from Specific IP Addresses or Ranges".
- Host access rules, where only specified host names, domain names, or subdomains are explicitly allowed or denied. This is analogous to Section 18.13.2.3, "Defining Access from a Specific Host or 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.

18.17.2. 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
```

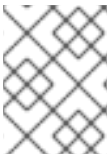
3. Restart the Directory Server to load the new plug-in configuration.

```
# systemctl restart dirsrv@instance
```

18.18. COMPATIBILITY WITH PREVIOUS RELEASES

For backward compatibility, the following deprecated ACI keywords are supported in Directory Server:

- **userdnattr**
- **groupdnattr**



NOTE

Red Hat recommends to not use these deprecated ACI keywords. These keywords will be removed from a future release of Directory Server.

CHAPTER 19. 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.

19.1. 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, *Managing Directory Entries*. For information on inactivating user accounts, see Section 19.15, "Manually Inactivating Users and Roles" .

Passwords can also be set and reset in the **Users and Groups** area of the Red Hat Administration Server or Directory Server Console. For information on how to use the **Users and Groups** area in the Administration Server Console, see the online help that is available in the Red Hat Administration Server.

Only password administrators, described in Section 19.2, "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.

19.2. 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 19.1, “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.

```
# 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=example,dc=com
changetype: modify
replace: passwordAdminDN
passwordAdminDN: cn=Passwd Admins,ou=groups,dc=example,dc=com
```

For setting in the global policy:

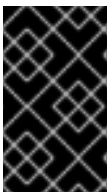
```
# ldapmodify  -h localhost -p 389 -D "cn=Directory Manager" -W
dn: cn=config
changetype: modify
replace: passwordAdminDN
passwordAdminDN: cn=Passwd Admins,ou=groups,dc=example,dc=com
```

19.3. 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 -p server_port -h server_hostname [-a oldPassword] [-s
newPassword] [user]
```



IMPORTANT

Password operations must be performed over a secure connection, meaning SASL, TLS, or Start TLS. For information on using secure connections with LDAP client tools, see Section 9.8.4, “Authenticating Using a Certificate” .

Table 19.1. ldappasswd Options

Parameter	Description
-h	Gives the host name of the Directory Server.

Parameter	Description
-p	Gives the port number of the Directory Server. Since TLS is required for password change operations, this is usually give the TLS 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 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 -p server_port -h server_hostname -Z [-a oldPassword] [-s newPassword] [user]
```

NOTE

For Start TLS connections to work, the TLS environment variables must be configured as described in Section 9.8.4, "Authenticating Using a Certificate" .

Use the **-ZZ** option 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 -p 389 -h server.example.com -x -ZZ -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.

19.4. 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 superseded by a subtree-level password policy, which is superseded 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.

19.4.1. Configuring the Global Password Policy

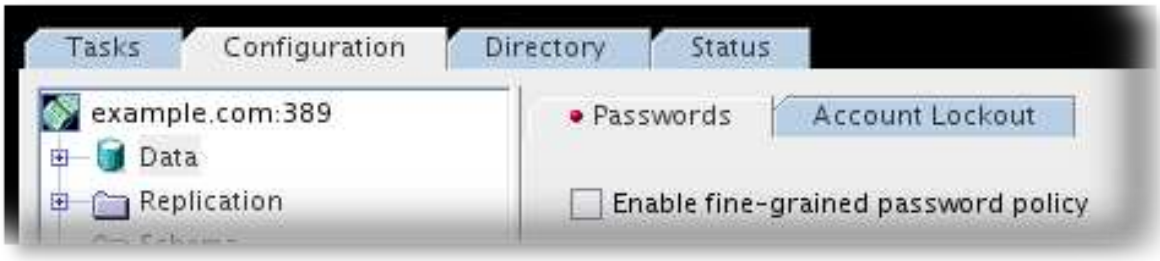
NOTE

After configuring the password policy, configure an account lockout policy. For details, see Section 19.8, "Configuring a Password-Based Account Lockout Policy" .

19.4.1.1. 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.
   - 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.

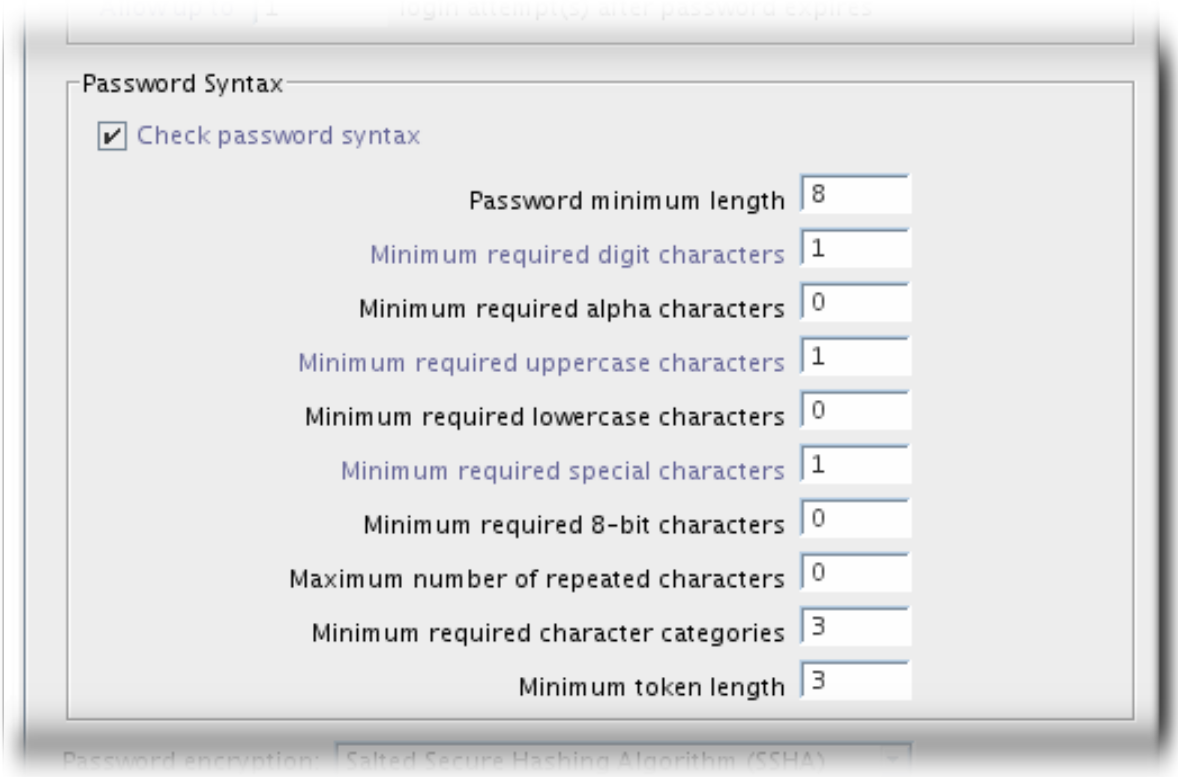
  The maximum value for the password age is derived by subtracting January 18, 2038, from today's date. The entered value must not be set to the maximum value or too close to the maximum value. Setting the value to the maximum value can cause the Directory Server to fail to start because the number of 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.

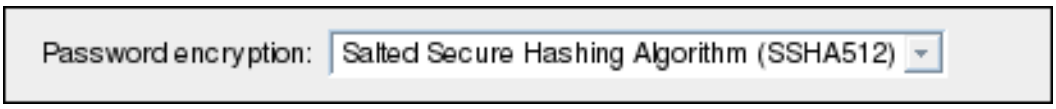
NOTE

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.



6. From the **Password Encryption** pull-down menu, select the encryption method for the server to use when storing passwords.



For a list of supported password storage schemes, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

7. Click **Save**.

19.4.1.2. 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
```

The following table displays the attributes you can use to configure the password policy:

Table 19.2. Password Policy-related Attributes

passwordChange	passwordCheckSyntax	passwordExp
passwordGraceLimit	passwordHistory	passwordInHistory
passwordMaxAge	passwordMaxRepeats	passwordMin8bit
passwordMinAge	passwordMinAlphas	passwordMinCategories
passwordMinDigits	passwordMinLength	passwordMinLowers
passwordMinSpecials	passwordMinTokenLength	passwordMinUppers
passwordMustChange	passwordSendExpiringTime	passwordStorageScheme
passwordTrackUpdateTime	passwordWarning	

For further details about the parameters, see the *Red Hat Directory Server Configuration, Command, and File Reference*.

2. Pass the LDIF file to the server using the **-f** option with the **ldapmodify** command.

```
# ldapmodify -D "cn=Directory Manager" -W -x -f user-pwdpolicy.ldif
```

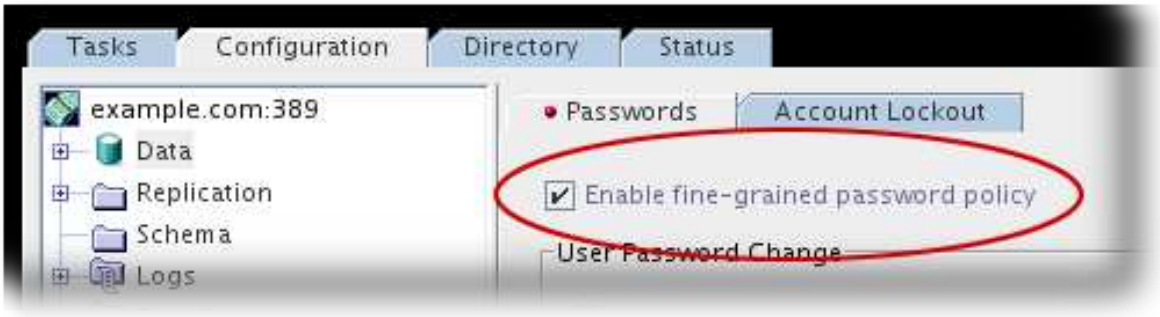
19.4.2. Configuring a Local Password Policy

NOTE

After configuring the password policy, configure an account lockout policy. For details, see Section 19.8, “Configuring a Password-Based Account Lockout Policy” .

19.4.2.1. Configuring a Subtree/User Password Policy Using the Console

1. Enable a fine-grained password policy globally, as described in Section 19.4.1.1, "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

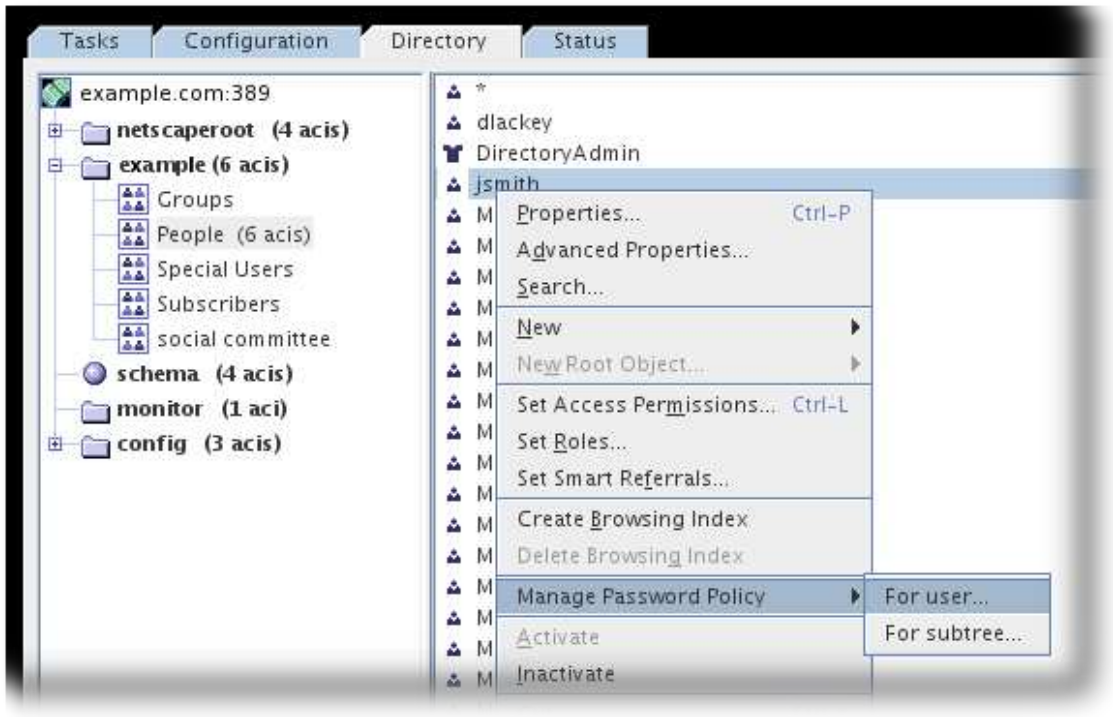
The global password policy does not override the local policy if they differ.

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 19.4.1.1, "Configuring a Global Password Policy Using the Console".

5. In the **Account Lockout** tab, specify the appropriate information, and click **Save**.

19.4.2.2. 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 *Red Hat Directory Server Configuration, Command, and File 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 19.2, "Password Policy-related Attributes" lists 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.

```
# systemctl stop dirsrv.target 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
```

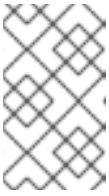
4. Start the server.

```
# systemctl start dirsrv.target instance
```

19.5. UNDERSTANDING PASSWORD EXPIRATION CONTROLS

When a user authenticates to Directory Server using a valid password, and if the password is expired, will expire soon, or needs to be reset, the server sends the following LDAP controls back to the client:

- Expired control (**2.16.840.1.113730.3.4.4**): Indicates that the password is expired. Directory Server sends this control in the following situations:
  - The password is expired, and grace logins have been exhausted. The server rejects the bind with an **Error 49** message.
  - The password is expired, but grace logins are still available. The bind will be allowed.
  - If ***passwordMustChange*** is enabled in the **cn=config** entry, and a user needs to reset the password after an administrator changed it. The bind is allowed, but any subsequent operation, other than changing the password, results in an **Error 53** message.
- Expiring control (**2.16.840.1.113730.3.4.5**): Indicates that the password will expire soon. Directory Server sends this control in the following situations:
  - The password will expire within the password warning period set in the ***passwordWarning*** attribute in the **cn=config** entry.
  - If the password policy configuration option is enabled in the ***passwordSendExpiringTime*** attribute in the **cn=config** entry, the expiring control is always returned, regardless of whether the password is within the warning period.
- Bind response control (**1.3.6.1.4.1.42.2.27.8.5.1**): The control contains detailed information about the state of the password that is about to expire or will expire soon.



NOTE

Directory Server only sends the bind response control if the client requested it. For example, if you use **ldapsearch**, you must pass the ***-e ppolicy*** parameter to the command to request the bind response control.

Example 19.1. Requesting the Bind Response Control in a Query

If you request the bind response control, for example by passing the ***-e ppolicy*** parameter to the **ldapsearch** command, the server returns detailed information about account expiration. For example:

```
# ldapsearch -D "uid=user_name,dc=example,dc=com" -xLLL -W \
    -b "dc=example,dc=com" -e ppolicy
ldap_bind: Success (0); Password expired (Password expired, 1 grace logins remain)
```

19.6. MANAGING THE DIRECTORY MANAGER PASSWORD

The Directory Manager is the privileged database administrator, comparable to the **root** user in Linux. The Directory Manager entry and the corresponding password are set during the instance installation.

The default distinguished name (DN) of the Directory Manager is **cn=Directory Manager**.

WARNING

Do not use curly braces (**{}**) in the password. Directory Server stores the password in the **{password-storage-scheme}hashed_password** format. The server interprets characters in curly braces as the password storage scheme. If the string is an invalid storage scheme or if the password is not correctly hashed, the Directory Manager cannot connect to the server.

19.6.1. Resetting the Directory Manager Password

If you lose the Directory Manager password, reset it:

1. Stop the Directory Server instance:

```
# systemctl stop dirsrv@instance_name
```

2. Generate a new password hash. For example:

```
# pwdhash -D /etc/dirsrv/slapd-instance_name password
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
UIhzo3LhiQLiv4iVGF7hEGezIka65kN
```

   Specifying the path to the Directory Server configuration automatically uses the password storage scheme set in the ***nsslapd-rootpwstoragescheme*** attribute to encrypt the new password.

3. Edit the **/etc/dirsrv/slapd-*instance_name*/dse.ldif** file and set the ***nsslapd-rootpw*** attribute to the value displayed in the previous step:

```
nsslapd-rootpw:
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
UIhzo3LhiQLiv4iVGF7hEGezIka65kN
```

4. Start the Directory Server instance:

```
# systemctl start dirsrv@instance_name
```

19.6.2. Changing the Directory Manager Password

19.6.2.1. Changing the Directory Manager Password Using the Command Line

To change the Directory Manager password using the command line, your server must support encrypted connections. If your server does not support encrypted connections, use the Directory Server Console to update the Directory Manager password. See Section 19.6.2.2, "Changing the Directory Manager Password Using the Directory Server Console".

If your server supports encrypted connections, perform these steps to change the password:

1. Generate a new password hash. For example:

```
# pwdhash -D /etc/dirsrv/slapd-instance_name password
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
Ulhzo3LhiQLiv4iVGF7hEGezIka65kN
```

   Specifying the path to the Directory Server configuration automatically uses the password storage scheme set in the ***nsslapd-rootpwstoragescheme*** attribute to encrypt the new password.

2. Set the ***nsslapd-rootpw*** attribute to the value displayed in the previous step using a secure connection (STARTTLS):

```
# ldapmodify -W -x -D "cn=Directory Manager" -p 389 -h server.example.com -x -ZZ

dn: cn=config
changetype: modify
replace: nsslapd-rootpw
nsslapd-rootpw:
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
Ulhzo3LhiQLiv4iVGF7hEGezIka65kN
```

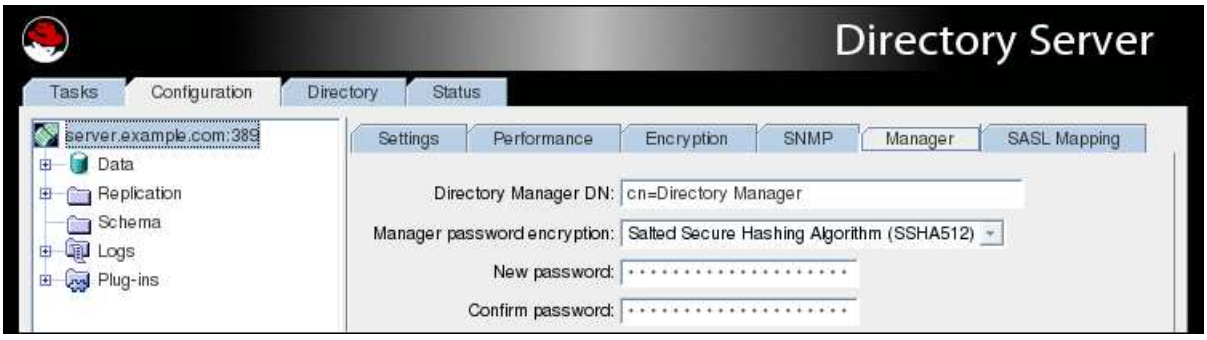
19.6.2.2. Changing the Directory Manager Password Using the Directory Server Console

As the administrator, perform these steps to change the password:

1. Open the Directory Server Console. For details, see Section 1.3.1, "Opening the Directory Server Console".

2. In the **Configuration** tab, select the host name in the left pane and click the **Manager** tab.

3. Enter a new password and confirm it.



4. Click **Save**.

19.6.3. Changing the Directory Manager Password Storage Scheme

The password storage scheme specifies which algorithm Directory Server uses to hash a password. To change the storage scheme using the command line, your server must support encrypted connections. If your server does not support encrypted connections, use the Directory Server Console to set the storage scheme. See Section 19.6.3.2, "Changing the Directory Manager Password Storage Scheme Using the Console".

Note that the storage scheme of the Directory Manager (***nsslapd-rootpwstoragescheme***) can be differ than the scheme used to encrypt user passwords (***nsslapd-pwstoragescheme***).

For a list of supported password storage schemes, see the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

NOTE

If you change the Directory Manager's password storage scheme you must also reset its password. Existing passwords cannot be re-encrypted.

19.6.3.1. Changing the Directory Manager Password Storage Scheme Using the Command Line

If your server supports encrypted connections, perform these steps to change the password storage scheme:

1. Generate a new password hash that uses the new storage scheme. For example:

```
# pwdhash -s SSHA512 password
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
Ulhzo3LhiQLiv4iVGF7hEGezIka65kN
```

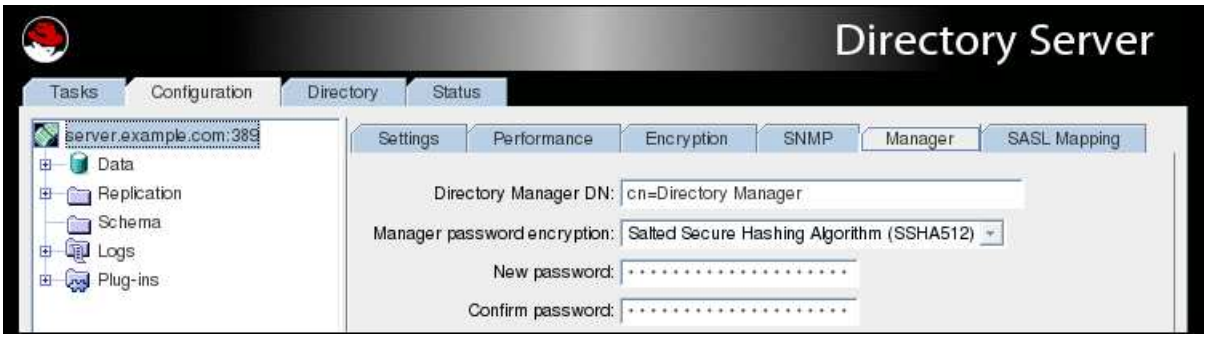
2. Set the ***nsslapd-rootpwstoragescheme*** attribute to the storage scheme and the ***nsslapd-rootpw*** attribute to the value displayed in the previous step using a secure connection (STARTTLS):

```
# ldapmodify -W -x -D "cn=Directory Manager" -p 389 -h server.example.com -x -F

dn: cn=config
changetype: modify
replace: nsslapd-rootpwstoragescheme
nsslapd-rootpwstoragescheme: SSHA512
-
replace: nsslapd-rootpw
nsslapd-rootpw:
{SSHA512}2eyW2uSFhh8LeB/nwZipfvFhSwL2DKZ58kXrCXsxr98Vz0nZI8fhd0W5BbL321Sr9
Ulhzo3LhiQLiv4iVGF7hEGezIka65kN
```

19.6.3.2. Changing the Directory Manager Password Storage Scheme Using the Console

As the administrator, perform these steps to change the Directory Manager password storage scheme:

1. Open the Directory Server Console. For details, see Section 1.3.1, "Opening the Directory Server Console".
2. In the **Configuration** tab, select to the host name in the left pane and click the **Manager** tab.
3. Select a new password storage scheme in the **Manager password encryption** field.



4. Enter a new password and confirm it.
5. Click **Save**.

19.6.4. Changing the Directory Manager DN

19.6.4.1. Changing the Directory Manager DN Using the Command Line

As the administrator, perform the following step to change the Directory Manager DN to **cn=New Directory Manager**:

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

dn: cn=config
changetype: modify
replace: nsslapd-rootdn
nsslapd-rootdn: cn=New Directory Manager
```

19.6.4.2. Changing the Directory Manager DN Using the Console

As the administrator, perform these steps to change the Directory Manager DN:

1. Open the Directory Server Console. For details, see Section 1.3.1, "Opening the Directory Server Console".
2. In the **Configuration** tab, select to the host name in the left pane and click the **Manager** tab.
3. Enter a new DN for the Directory Manager into the **Directory Manager DN** field.

4. Click **Save**.

19.7. 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.

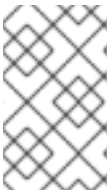
There are some situations where a client requires information about a user account – specifically whether an account should be allowed to authenticate – in order to perform some other operation, but the client either does not have or does use any credentials for the user account in Directory Server. 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.

19.7.1. 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:

```
# 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:

```
# 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 19.7.2, “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 19.3. 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
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

Account Status	Control Result Message
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

19.7.2. 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, to enable members of the **cn=Administrators,ou=groups,dc=example,dc=com** group to read the Account Usability Extension Control of all users:

```
# 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 = "*")(version 3.0; acl "Account Usable"; allow (read)(groupdn =
"ldap:///cn=Administrators,ou=groups,dc=example,dc=com");)
```

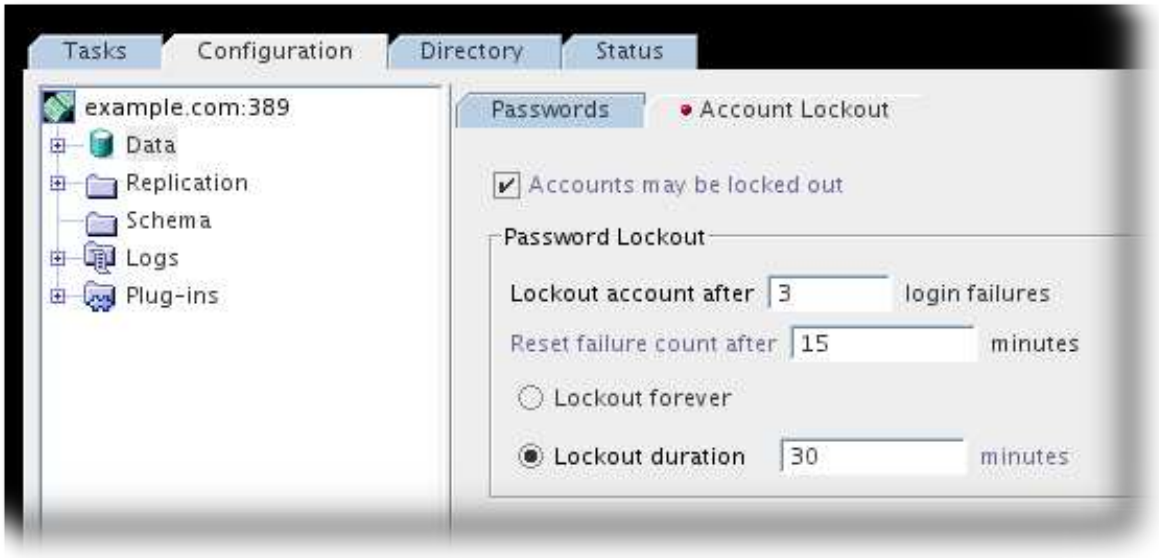
19.8. CONFIGURING A PASSWORD-BASED ACCOUNT LOCKOUT POLICY

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.

19.8.1. 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**.

19.8.2. Configuring the Account Lockout Policy Using the Command Line

Use **ldapmodify** to configure account lockout policy settings in the **cn=config** entry. For example:

```
# 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
-
```

Attributes related to account lockout policy are described in the *Red Hat Directory Server Configuration, Command, and File Reference*.

The following attributes control the account password policy:

- passwordLockout
- passwordMaxFailure
- passwordUnlock
- passwordLockoutDuration
- passwordResetFailureCount

19.8.3. 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" -x -D "cn=directory manager" -W -p 389 -h
server.example.com -x
dn: cn=config
replace: passwordLegacyPolicy
passwordLegacyPolicy: off
```

19.9. 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.

19.9.1. 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 described in the *Account Policy Plug-in Attributes* section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

- 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 described in the following table:

  Table 19.4. 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

For further details, see the attribute's description in the *Red Hat Directory Server Configuration, Command, and File Reference*

19.9.2. Account Inactivity and Account Expiration

The **Account Policy** plug-in enables you to set up:

- account expiration: Accounts are disabled a certain amount of time after you created an account.
- account inactivity: Accounts are disabled a certain amount of time after the last successful login. This enables you to automatically disable unused accounts.

Disabled accounts are no longer able to log in.

To set up the **Account Policy** plug-in:

1. Enable the Account Policy Plug-in.

```
# 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.

```
# 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.

   - 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**).

```
# 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.

```
# systemctl start dirsrv.target
```

5. Define an account policy.

```
# 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.

```
# 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.

```
# 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
```

19.9.3. Disabling Accounts a Certain Amount of Time After Password Expiry

Directory Server enables you to configure an account policy that disables an account a certain amount of time after the password expired. Disables accounts are no longer able to log in.

To set up this configuration, follow the procedure in Section 19.9.2, "Account Inactivity and Account Expiration". However, when configuring the plug-in configuration entry, use the following settings instead:

```
dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

objectClass: top
objectClass: extensibleObject
cn: config
alwaysrecordlogin: yes
stateAttrName: non_existent_attribute
altStateAttrName: passwordExpirationTime
specattrname: acctPolicySubentry
limitattrname: accountInactivityLimit
```

This configuration uses a dummy value in the ***stateAttrName*** parameter. Therefore, only the **passwordExpirationTime** attribute set in the ***altStateAttrName*** parameter is used to calculate when an account is expired.

To additionally record the time of the last successful login in the ***lastLoginTime*** attribute of the user entry, set:

```
dn: cn=config,cn=Account Policy Plugin,cn=plugins,cn=config

alwaysRecordLoginAttr: lastLoginTime
```

Using this configuration, an account is automatically disabled if the sum of the time set in the user's ***passwordExpirationTime*** attribute and in the ***accountInactivityLimit*** parameter's value is in the past. Using this configuration, an account is automatically disabled if the sum of the value in the user's ***passwordExpirationTime*** attribute and in the ***accountInactivityLimit*** parameter exceeds the time since the ***alwaysRecordLoginAttr*** attribute was last updated.

19.9.4. 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.

```
# 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.

```
# 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***).

```
# 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.

```
# systemctl start dirsrv.target
```

19.9.5. 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:

```
# 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.

19.10. 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***

19.10.1. 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***

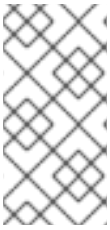
However, the configuration information is kept locally and is not replicated. This information includes the password syntax and the history of password modifications. Account lockout counters and tiers are not replicated, either, unless specifically configured for replication.

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.

19.10.2. 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:

```
# 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.

19.10.3. 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.

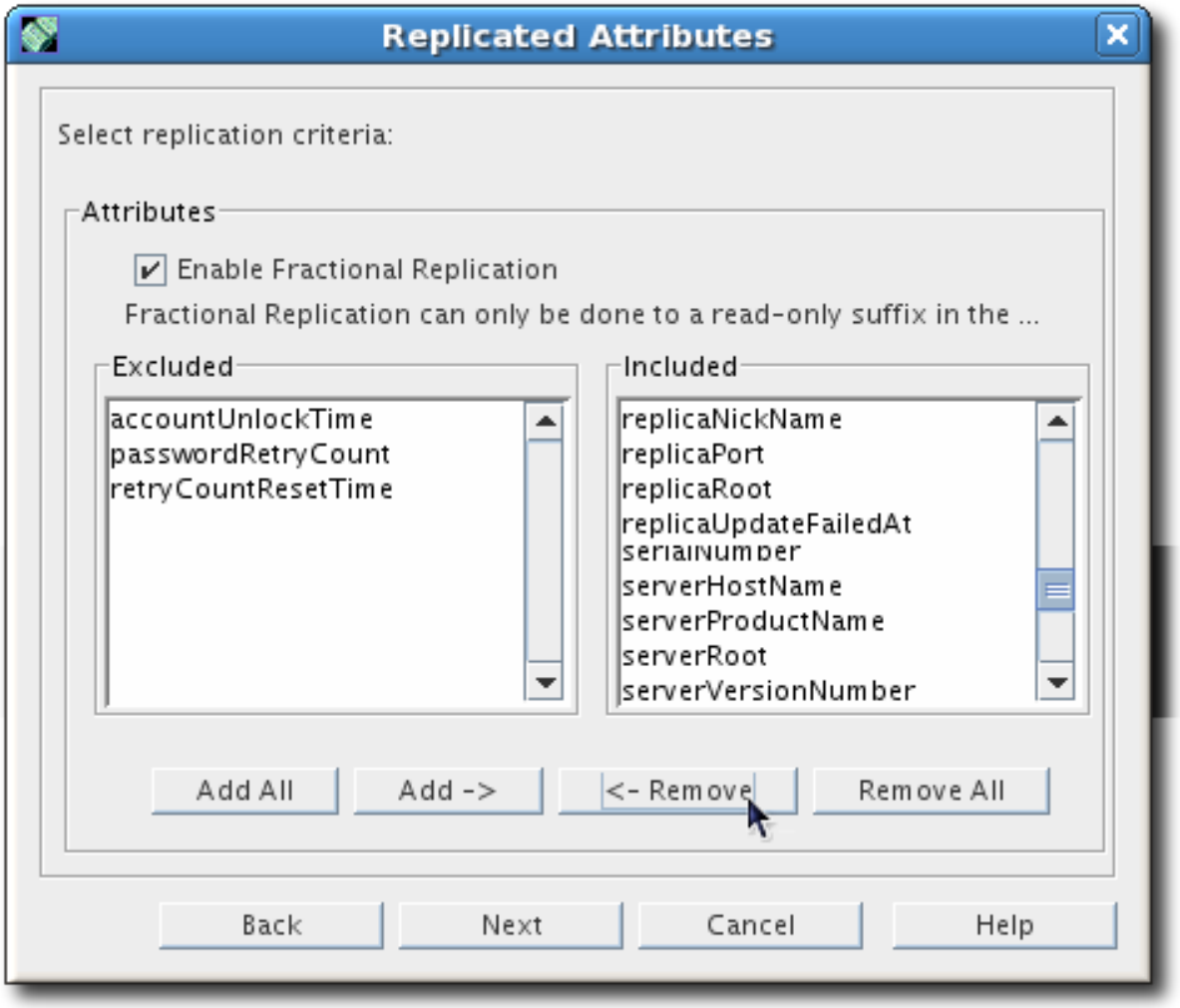
If the password policy attributes should be replicated, then make sure these attributes are included in the fractional replication agreement (as they are by default).

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 15.1.7, “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 15.5.3, "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.

19.11. 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 19.11.1, "Requiring Secure Binds") or streamline bind operations (such as Section 19.11.4, "Configuring Autobind").

19.11.1. 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 (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 agreements, 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:

```
# 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.

```
# systemctl restart dirsrv.target
```

19.11.2. 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 18.13.1.1.3, "Granting Anonymous Access" has an example on setting ACIs to control what anonymous users can access, and Section 14.1.5, "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:

```
# 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.

```
# systemctl restart dirsrv.target
```

NOTE

With anonymous binds disabled, the users cannot log in using their RDN. They are required to provide the full DN to log in.

In addition, when you disable anonymous binds, unauthenticated binds are also disabled automatically.

19.11.3. Allowing Unauthenticated Binds

Unauthenticated binds are connections to Directory Server where a user supplies an empty password. Using the default settings, Directory Server denies access in this scenario for security reasons:

```
# ldapsearch -w "" -p 389 -h server.example.com -b "dc=example,dc=com" \
    -s sub -x "(objectclass=*)"

ldap_bind: Server is unwilling to perform (53)
 additional info: Unauthenticated binds are not allowed
```

WARNING

Red Hat recommends not enabling unauthenticated binds. This authentication method enables users to bind without supplying a password as any account, including the Directory Manager. After the bind, the user can access all data with the permissions of the account used to bind.

To enable insecure unauthenticated binds, set the ***nsslapd-allow-unauthenticated-binds*** to **on**:

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

dn: cn=config
changetype: modify
replace: nsslapd-allow-unauthenticated-binds
nsslapd-allow-unauthenticated-binds: on
```

19.11.4. 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 1.6, "Enabling LDAPI"). Then, configure the autobind mappings (in Section 19.11.4.2, "Configuring Autobind").

19.11.4.1. 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**

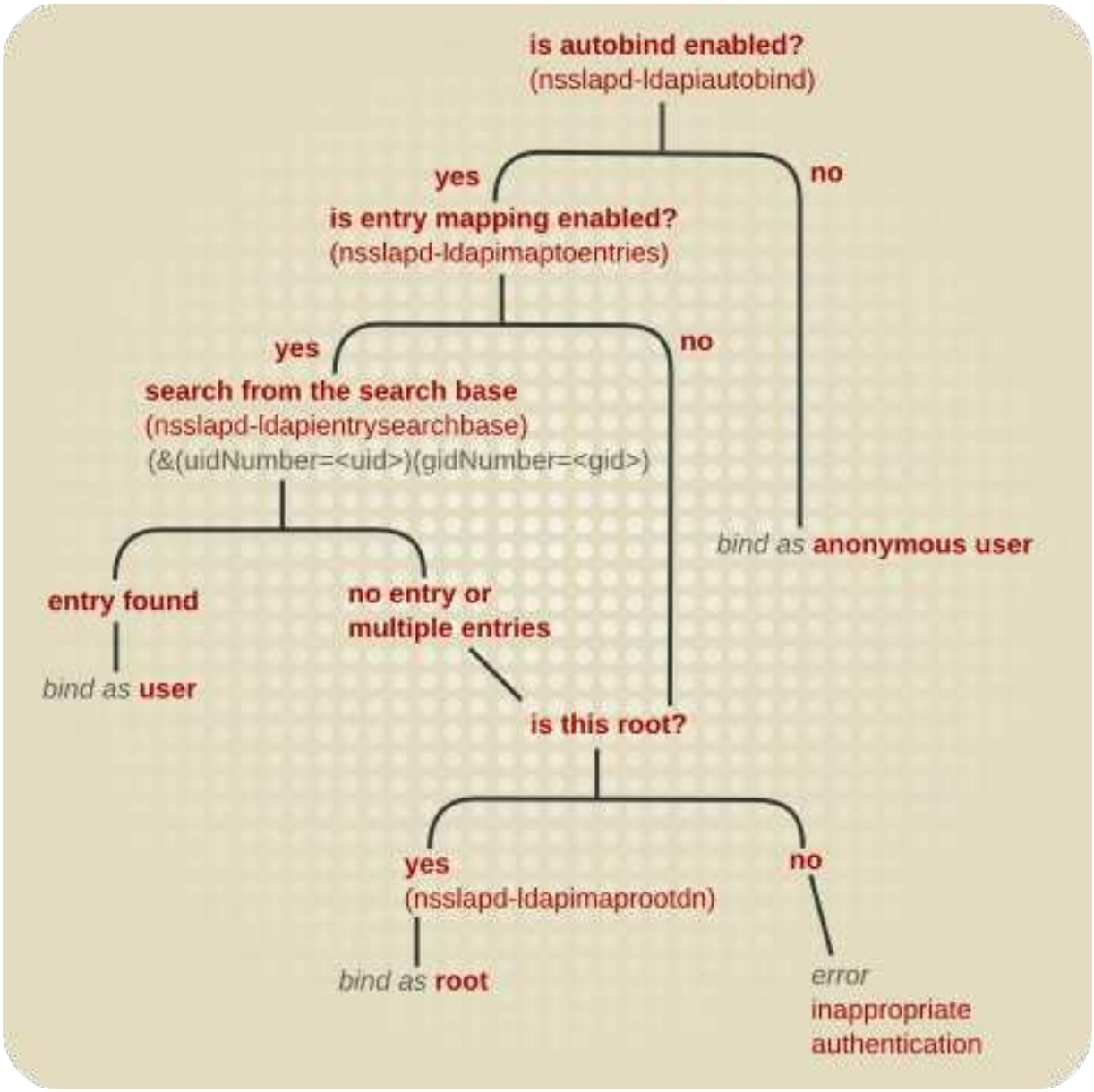


Figure 19.1. Autobind Connection Path

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*+uidNumber*uid*, *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 9.9, “Setting up SASL Identity Mapping”.

19.11.4.2. 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.

```
# systemctl restart dirsrv@instance
```

19.12. 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.

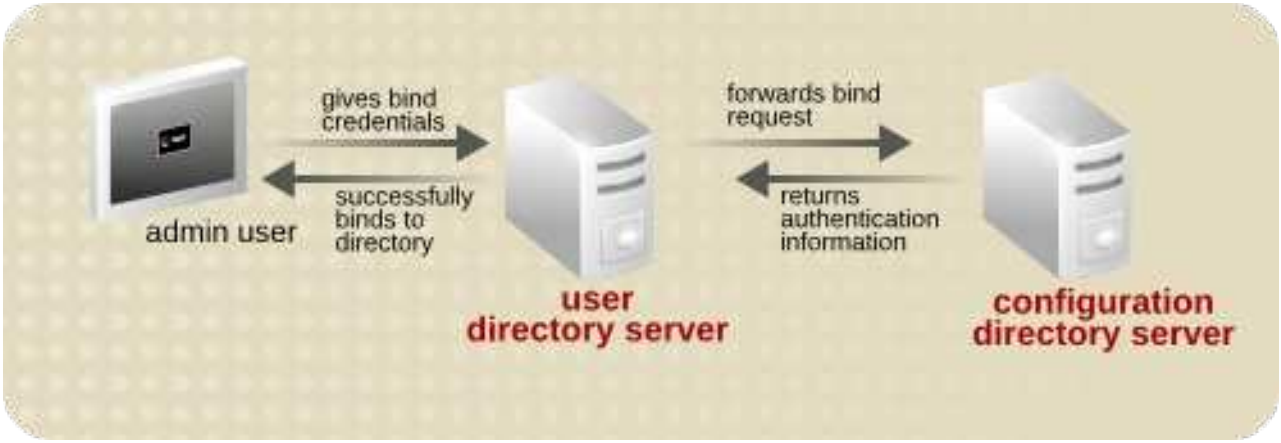


Figure 19.2. Simple Pass-Through Authentication Process

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.

19.12.1. 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 19.5, “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 19.12.3.2, "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 19.5. 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 TLS is used for communication between the two Directory Servers. See Section 19.12.2.1, "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 19.12.2.2, "Specifying the Authenticating Directory Server" for more information.
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 19.12.2.3, "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 19.12.2.4, "Configuring the Optional Parameters" for more information.

Variable	Definition
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 19.12.2.4, "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 19.12.2.4, "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 19.12.2.4, "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 19.12.2.4, "Configuring the Optional Parameters" for more information.
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 TLS 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 19.12.2.4, "Configuring the Optional Parameters" for more information.

19.12.2. 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 19.12.1, "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 19.12.2.1, "Configuring the Servers to Use a Secure Connection"
- Section 19.12.2.2, "Specifying the Authenticating Directory Server"
- Section 19.12.2.3, "Specifying the Pass-Through Subtree"
- Section 19.12.2.4, "Configuring the Optional Parameters"

19.12.2.1. Configuring the Servers to Use a Secure Connection

The PTA directory can be configured to communicate with the authenticating directory over TLS by specifying LDAPS in the LDAP URL of the PTA directory. For example:

```
nsslapd-pluginarg0: ldaps://ldap.example.com:636/o=NetscapeRoot
```

19.12.2.2. 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 19.5, "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.

```
systemctl restart dirsrv@instance
```

19.12.2.3. 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 19.5, “PTA Plug-in Parameters”.

2. Restart the server.

```
# systemctl restart dirsrv@instance
```

19.12.2.4. 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 TLS.

  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.

```
# systemctl restart dirsrv@instance
```

19.12.3. PTA Plug-in Syntax Examples

This section contains the following examples of PTA Plug-in syntax in the **dse.ldif** file:

- Section 19.12.3.1, "Specifying One Authenticating Directory Server and One Subtree"
- Section 19.12.3.2, "Specifying Multiple Authenticating Directory Servers"
- Section 19.12.3.3, "Specifying One Authenticating Directory Server and Multiple Subtrees"
- Section 19.12.3.4, "Using Non-Default Parameter Values"
- Section 19.12.3.5, "Specifying Different Optional Parameters and Subtrees for Different Authenticating Directory Servers"

19.12.3.1. Specifying One Authenticating Directory Server and One Subtree

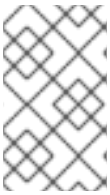
This example configures the PTA Plug-in to accept all defaults for the optional variables. This configuration causes the PTA Directory Server to connect to the authenticating Directory Server for all bind requests to the **o=NetscapeRoot** subtree. The host name of the authenticating Directory Server is **configdir.example.com**.

```
dn: cn=Pass Through Authentication,cn=plugins,cn=config
...
nsslapd-pluginEnabled: on
nsslapd-pluginarg0: ldap://configdir.example.com/o=NetscapeRoot
...
```

19.12.3.2. 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*.

19.12.3.3. 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
...
```

19.12.3.4. 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
...
```

19.12.3.5. 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
...
```

19.13. USING ACTIVE DIRECTORY-FORMATTED USER NAMES FOR AUTHENTICATION

When you connect to Directory Server, you must specify the distinguished name (DN) of the user, such as **uid=*user_name*,ou=People,dc=example,dc=com**, to authenticate. However, the DN can be difficult to remember. If you enable and configure the **AD DN** plug-in, you can use Active Directory-formatted user names, such as ***user_name*** or ***user_name@domain*** instead of the DN.

After you enable the plug-in and a user connects to the directory using a user name that is not DN-formatted, Directory Server searches the DN based on the plug-in's configuration. If the search returns one DN, Directory Server uses this DN for the authentication. If none or multiple DNs are returned, authentication fails.

NOTE

You can only enable and configure the **AD DN** plug-in using the command line.

To enable and configure the plug-in it to use **example.com** as the default domain:

1. Add the **cn=addn,cn=plugins,cn=config** plug-in entry and set the default domain:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=addn,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: nsSlapdPlugin
objectClass: extensibleObject
cn: addn
nsslapd-pluginPath: libaddn-plugin
nsslapd-pluginInitfunc: addn_init
nsslapd-pluginType: preoperation
nsslapd-pluginEnabled: on
nsslapd-pluginId: addn
nsslapd-pluginVendor: 389 Project
nsslapd-pluginVersion: 1.3.6.0
nsslapd-pluginDescription: Allow AD DN style bind names to LDAP
addn_default_domain: example.com
```

The required ***addn_default_domain*** parameter in the plug-in entry sets the default domain. The plug-in appends this domain if the specified user name during an authentication does not contain a domain name.

2. Add a configuration entry for the default domain:

```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x
dn: cn=example.com,cn=addn,cn=plugins,cn=config
changetype: add
objectClass: top
objectClass: extensibleObject
cn: example.com
addn_base: ou=People,dc=example,dc=com
addn_filter: (&(objectClass=account)(uid=%s))
```

For details about the parameters used in the example, see their descriptions in the *Red Hat Directory Server Configuration, Command, and File Reference* .

WARNING

You must add at least a configuration entry for the default domain. If the entry is missing, Directory Server fails to start.

3. Optionally, you can create additional domain configurations as described in the previous step to support multiple domain names. Each domain configuration can use a different search base and filter.

4. Restart the Directory Server instance:

```
# systemctl restart dirsrv@instance_name
```

19.14. 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.

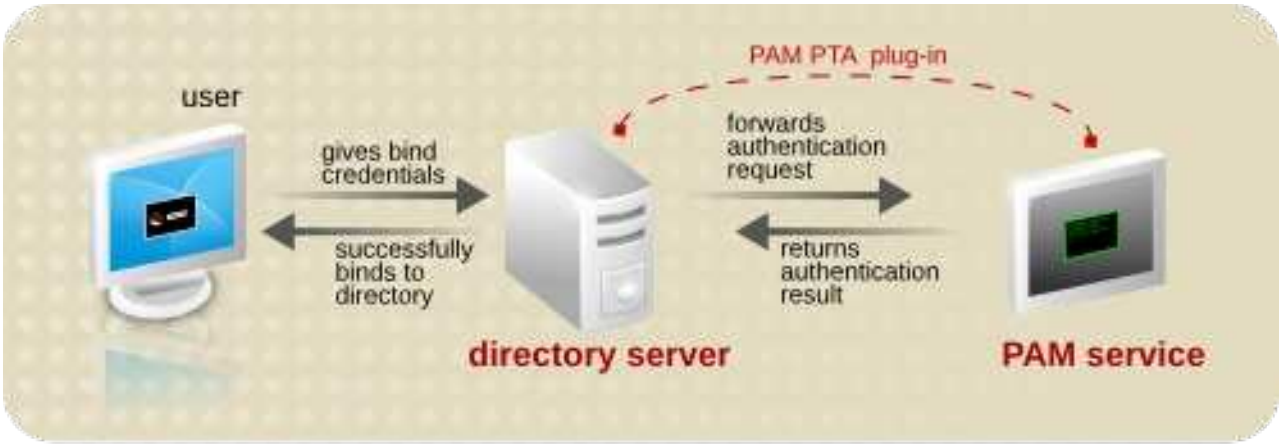
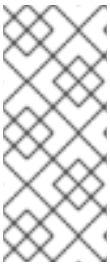


Figure 19.3. PAM Pass-Through Authentication Process



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.

19.14.1. 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 TLS 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.

For a list of attributes you can set, see the *PAM Pass Through Auth Plug-in Attributes* section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

19.14.1.1. 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
```

19.14.1.2. 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.

19.14.1.3. Setting PAM PTA Mappings

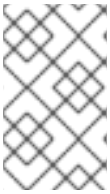
There has to be a way to connect the LDAP identity to the PAM identity. The first thing to define is the *method* to use to map the entries. There are three options: DN, RDN, and ENTRY. ENTRY uses a user-defined attribute in the entry.

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 19.6, "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 19.6. Mapping Methods for PAM Authentication

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. For example: **pamIDAttr: customPamUid**
DN	This method uses the full distinguished name from the bind DN. The mapping for this method is defined by Directory Server.

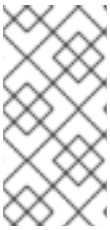
19.14.1.4. Configuring General PAM PTA Settings

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
```

19.14.2. 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 Section 19.14.1, “PAM Pass-Through Authentication Configuration Options” , and Example 19.2, “Example PAM Pass-Through Authentication Configuration Entry” has an example entry.
6. Restart the server to load the new plug-in configuration.

```
# systemctl restart dirsrv.target
```

Example 19.2. 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
```

19.14.3. 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.

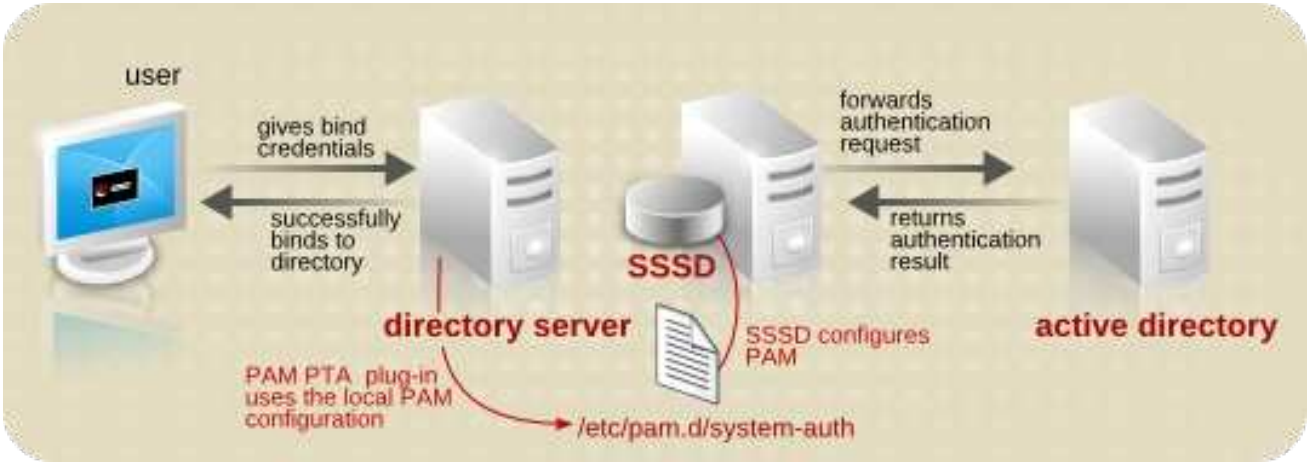


Figure 19.4. PAM Pass-Through Authentication with SSSD

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 *Using Active Directory as an Identity Provider for SSSD* section in the *Windows Integration Guide*.

2. Enable the PAM Pass-Through Auth plug-in; this is disabled by default.

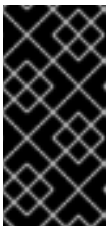
```
# ldapmodify -D "cn=Directory Manager" -W -p 389 -h server.example.com -x

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 synchronized, 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.

```
# systemctl restart dirsrv.target
```

19.15. 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.

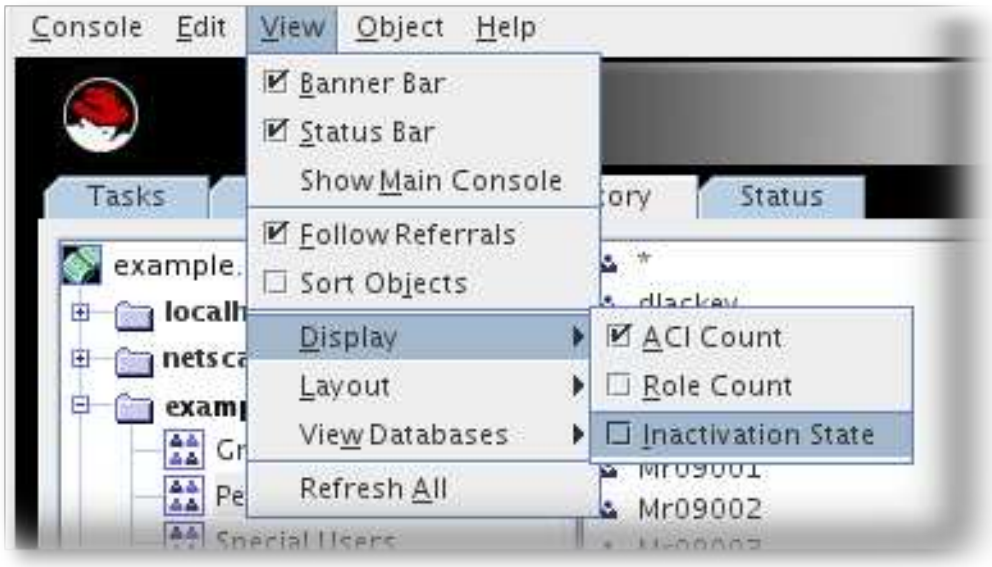
The same procedures are used to inactivate users and roles. However, when a role is inactivated, the *members of the role* are inactivated, not the role entry itself. For more information about roles in general and how roles interact with access control in particular, see Chapter 8, *Organizing and Grouping Entries*.

WARNING

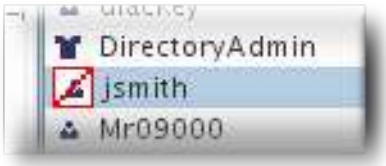
The root entry (the entry corresponding to the root or sub suffix) on a database cannot be inactivated. Chapter 3, *Managing Directory Entries* has information on creating the entry for a root or sub suffix, and Chapter 2, *Configuring Directory Databases* has information on creating root and sub suffixes.

19.15.1. Viewing Inactive Users and Roles Using the Console

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.

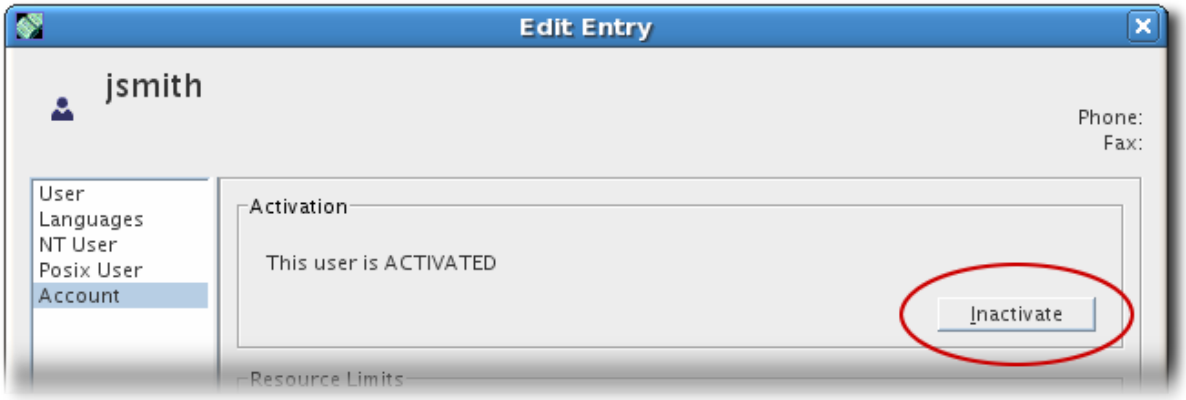


19.15.2. Activating and Inactivating Users and Roles Using the Console

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.

19.15.3. Viewing Inactive Users and Roles Using the Command Line

The **ns-accountstatus.pl** script is used to obtain detailed information about active and inactive users.

To obtain the account status of a single user, you can use the command as follows:

```
# ns-accountstatus.pl -D "cn=Directory Manager" -w password -I
"uid=jsmith,ou=people,dc=example,dc=com"
uid=bjensen,ou=people,dc=example,dc=com  activated.
```

Add the **-V** option to obtain more verbose output:

```
# ns-accountstatus.pl -D "cn=Directory Manager" -w password -I
"uid=jsmith,ou=people,dc=example,dc=com"
Entry:                  uid=jsmith,ou=People,dc=example,dc=com
Entry Creation Date:    20160204153140Z (02/04/2016 10:31:40)
Entry Modification Date: 20160205163904Z (02/05/2016 11:39:04)
Last Login Date:        20160205163905Z (02/05/2016 11:39:05)
Inactivity Limit:       2592000 seconds (30 days)
Time Until Inactive:    2591688 seconds (29 days, 23 hours, 54 minutes, 48 seconds)
Time Since Inactive:    -
Entry State:            activated
```

The above is an example of an active account, as indicated by the last three lines of the output. An inactivated account would instead provide output similar to the following:

-

```
# ns-accountstatus.pl -D "cn=Directory Manager" -w password -I
"uid=jsmith,ou=people,dc=example,dc=com"
Entry:                  uid=jsmith,ou=people,dc=example,dc=com
Entry Creation Date:    20160204153140Z (02/04/2016 10:31:40)
Entry Modification Date: 20160204160545Z (02/04/2016 11:05:45)
Last Login Date:        20160204160546Z (01/04/2016 11:05:46)
Inactivity Limit:       2592000 seconds (30 days)
Time Until Inactive:    -
Time Since Inactivated:  85877 seconds (23 hours, 51 minutes, 17 seconds)
Entry State:            inactivated (inactivity limit exceeded)
```

Instead of using the **-I** option to specify an account, you can use the **-b** (search a database suffix), **-f** (use a filter), and **-s** (search scope) options to create a search. Additionally, you can refine the search by using the **-i** option (return only inactive accounts) or the **-g *X*** option (return only accounts which will expire in the next *X* seconds). For example:

```
# ns-accountstatus.pl -D "cn=Directory Manager" -w password -b "ou=people,dc=example,dc=com" -f
"(uid=*)" -V -g 86400
Entry:                  uid=jsmith,ou=people,dc=example,dc=com
Entry Creation Date:    20160204153140Z (02/04/2016 10:31:40)
Entry Modification Date: 20160205163904Z (02/05/2016 11:39:04)
Last Login Date:        20160205163905Z (01/05/2016 11:39:05)
Inactivity Limit:       2592000 seconds (30 days)
Time Until Inactive:    979 seconds (16 minutes, 19 seconds)
Time Since Inactive:    -
Entry State:            activated
```

As you can see from the last three lines of the output, this account is currently active, but will expire soon.

19.15.4. 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 the *Red Hat Directory Server Configuration, Command, and File Reference*.

For example, to inactivate a user account:

```
[root@server ~]# ns-inactivate.pl -Z instance_name -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:

```
# ns-activate.pl -Z instance_name -D Directory Manager -w secret -p 389 -h example.com  -I
"uid=jfrasier,ou=people,dc=example,dc=com"
```

CHAPTER 20. 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 21, *Monitoring Directory Server Using SNMP*.

20.1. TYPES OF DIRECTORY SERVER LOG FILES

Directory Server provides the following log types:

- Access log: Contains information on client connections and connection attempts to the Directory Server instance. This log type is enabled by default.
- Error log: Contains detailed messages of errors and events the directory experiences during normal operations. This log type is enabled by default.

WARNING

If the Directory Server fails to write to the errors log, the server sends an error message to the **Syslog** service and exits. This log type is enabled by default.

- Audit log: Records changes made to each database as well as to server configuration. This log is not enabled by default.
- Audit fail log: Records failed audit events. This log is not enabled by default.

20.2. DISPLAYING LOG FILES

You can display the Directory Server log files using the command line and Directory Server Console.

20.2.1. Displaying Log Files Using the Command Line

To display the log files using the command line, use the utilities included in Red Hat Enterprise Linux, such as **less**, **more**, and **cat**. For example:

```
# less /var/log/dirsrv/slapd-instance_name/access
```

To display the locations of log files:

```
# ldapsearch -D "cn=Directory Manager" -W -p 389 \
    -h server.example.com -x -b "cn=config" -s base \
    nsslapd-accesslog nsslapd-errorlog nsslapd-auditlog nsslapd-auditfaillog
...
nsslapd-accesslog: /var/log/dirsrv/slapd-instance_name/access
nsslapd-errorlog: /var/log/dirsrv/slapd-instance_name/errors
nsslapd-auditlog: /var/log/dirsrv/slapd-instance_name/audit
nsslapd-auditfaillog: /var/log/dirsrv/slapd-instance_name/audit-failure
```

-

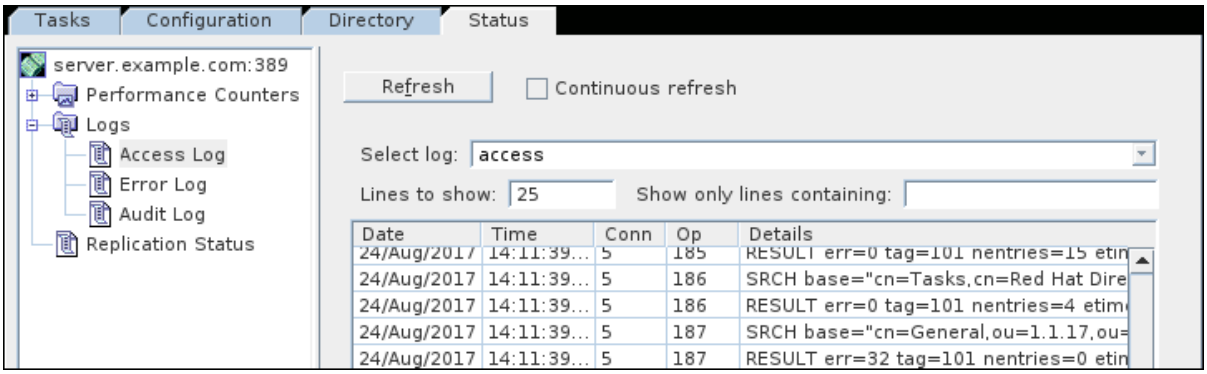
NOTE

If logging for a log type is not enabled, the corresponding log file does not exist.

20.2.2. Displaying Log Files Using the Console

To display the Directory Server log files:

1. Open the Directory Server Console. For details, see Section 1.3.1, "Opening the Directory Server Console".
2. Select the **Status** tab.
3. Expand the **Logs** entry and select the log you want to display.



NOTE

The Console does not contain a log file viewer for the **Audit Fail** log. Alternatively, you can:

- Display this log using the command line. See Section 20.2.1, "Displaying Log Files Using the Command Line".
- Configure Directory Server to log **Audit Fail** entries to the same file as **Audit** events.

4. Optionally, you can apply the following settings to the log file viewer:
   - Set the number of line to display in the **Lines to show** field.
   - Set a filter in the **Show only lines containing** field.
   - Display older log files of the same type, by selecting the log in the **Select log** field.
   - Enable automatically displaying new log entries by selecting **Continuous refresh**.

   Click the **Refresh** button to apply the changes.

20.3. 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.

20.3.1. Enabling or Disabling Logs

The access and error logging is enabled by default. However, audit and audit fail 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***
- Audit fail log: ***nsslapd-auditfaillog-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:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-auditlog-logging-enabled
nsslapd-auditlog-logging-enabled: on
```

20.3.2. Configuring Plug-in-specific Logging

For debugging, you can enable access and audit logging for operations a plug-ins executes. For details, see the **nsslapd-logAccess** and **nsslapd-logAudit** parameter in the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference* .

20.3.3. Disabling High-resolution Log Time Stamps

Using the default settings, Directory Server logs entries with nanosecond precision:

```
[27/May/2016:17:52:04.754335904 -0500] schemareload - Schema validation passed.
[27/May/2016:17:52:04.894255328 -0500] schemareload - Schema reload task finished.
```

To disable high-resolution log time stamps:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-logging-hr-timestamps-enabled
nsslapd-logging-hr-timestamps-enabled: off
```

NOTE

The option to disable high-resolution log time stamps is deprecated and will be removed in a future release.

After disabling high-resolution log time stamps, Directory Server logs with second precision only:

```
[27/May/2016:17:52:04 -0500] schemareload - Schema validation passed.
[27/May/2016:17:52:04 -0500] schemareload - Schema reload task finished.
```

20.3.4. 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***
- Audit fail log: ***nsslapd-auditfaillog-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***
- Audit fail log: ***nsslapd-auditfaillog-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***
- Audit fail log: ***nsslapd-auditfaillog-maxlogsize***

Create a log every

Sets the maximum age of a log file.

- ***nsslapd-accesslog-logrotationtime*** and ***nsslapd-accesslog-logrotationtimeunit***
- ***nsslapd-errorlog-logrotationtime*** and ***nsslapd-errorlog-logrotationtimeunit***
- ***nsslapd-auditlog-logrotationtime*** and ***nsslapd-auditlog-logrotationtimeunit***
- ***nsslapd-auditfaillog-logrotationtime*** and ***nsslapd-auditfaillog-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***

- ***nsslapd-auditfaillog-logrotationsynchour*** and ***nsslapd-auditfaillog-logrotationsyncmin***

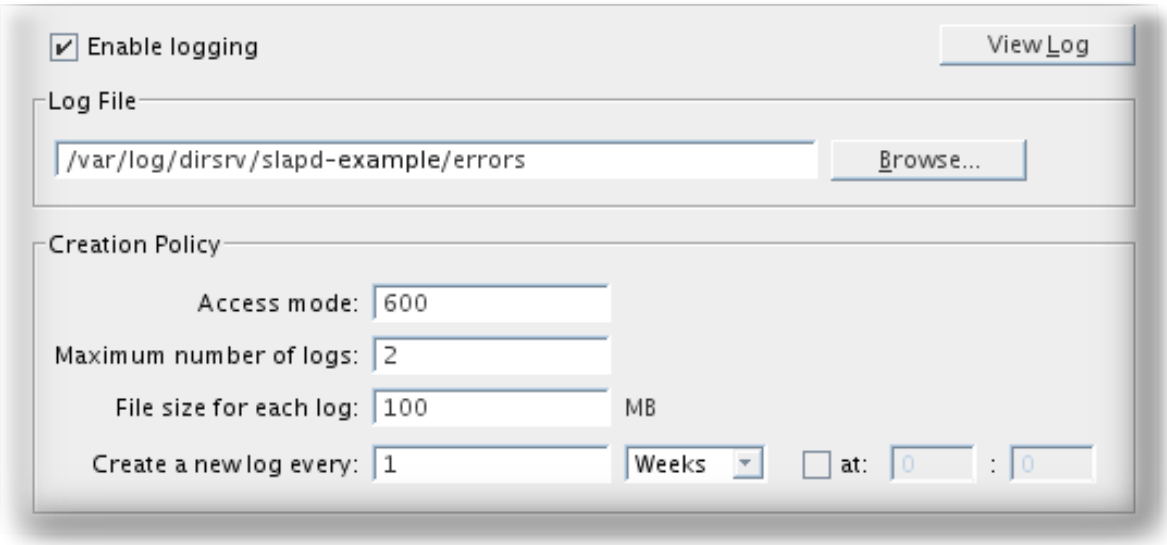
For details, see the parameter descriptions in the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

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/1.3.5.10 B2016.257.1817
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:

```
# 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
```

20.3.5. 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, audit or audit fail 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***
- Audit log: ***nsslapd-auditfaillog-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***
- Audit log: ***nsslapd-auditfaillog-logminfreediskspace***

When a file is older than a specified time

When a log file is older than the configured time, it is automatically deleted.

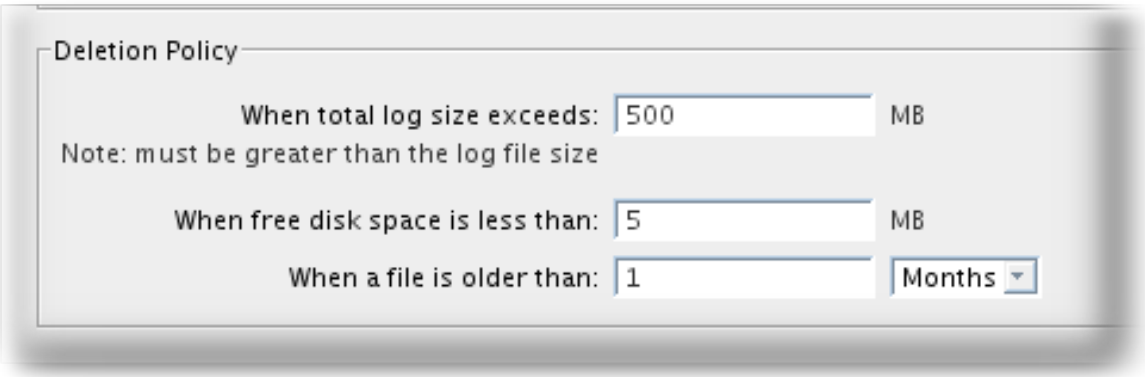
- Access log: ***nsslapd-accesslog-logexpirationtime*** and ***nsslapd-accesslog-logexpirationtimeunit***
- Error log: ***nsslapd-errorlog-logminfreediskspace*** and ***nsslapd-errorlog-logexpirationtimeunit***
- Audit log: ***nsslapd-auditlog-logminfreediskspace*** and ***nsslapd-auditlog-logexpirationtimeunit***

- Audit log: ***nsslapd-auditfaillog-logminfreediskspace*** and ***nsslapd-auditfaillog-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:

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-accesslog-logmaxdiskspace
nsslapd-accesslog-logmaxdiskspace: 500
```

20.3.6. 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, audit and audit fail 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
```

20.3.7. 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.

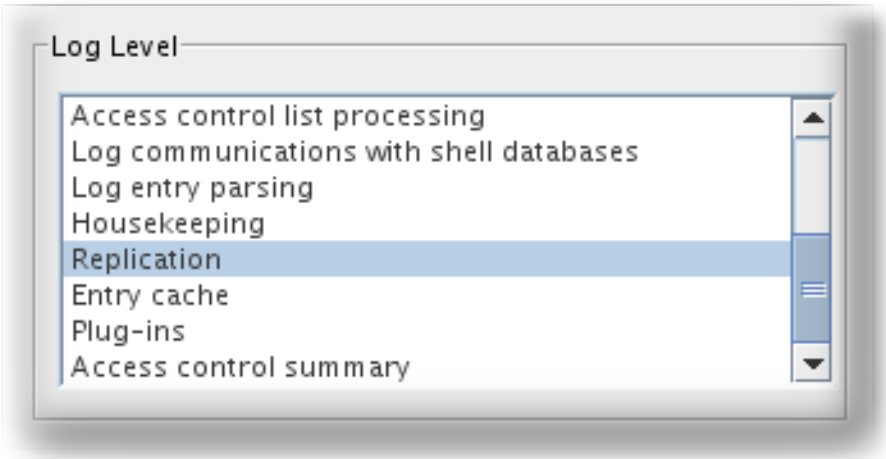
Configuring the Log Level in the Directory Server Console

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):

```
# ldapmodify -D "cn=Directory Manager" -W -x
dn: cn=config
changetype: modify
replace: nsslapd-errorlog-level
nsslapd-errorlog-level: 96
```

20.4. 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, Command, and File 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 8.2
Command: logconv.pl /var/log/dirsrv/slapd-instance/access
Processing 1 Access Log(s)...

[001] /var/log/dirsrv/slapd-instance/access size (bytes):       77532

Total Log Lines Analysed:  527

Start of Logs:    14/Oct/2017:16:15:22.452909568
End of Logs:      14/Oct/2017:16:39:50.157790196

Processed Log Time:  0 Hours, 24 Minutes, 27.704877056 Seconds

Restarts:                   10
Secure Protocol Versions:
  - TLS1.2 client bound as uid=user_name,ou=people,o=example.com (11 connections)
  - TLS1.2 128-bit AES; client CN=CA Subsystem,O=example.com; issuer CN=Certificate
Authority,O=example.com (11 connections)
  - TLS1.2 128-bit AES-GCM (2 connections)
```

```
  - TLS1.2 128-bit AES (3 connections)

Peak Concurrent Connections:  38
Total Operations:             4771
Total Results:                4653
Overall Performance:          97.5%

Total Connections:            249          (0.17/sec)  (10.18/min)
 - LDAP Connections:          107          (0.07/sec)  (4.37/min)
 - LDAPI Connections:         128          (0.09/sec)  (5.23/min)
 - LDAPS Connections:         14           (0.01/sec)  (0.57/min)
 - StartTLS Extended Ops:     2            (0.00/sec)  (0.08/min)

Searches:                     2963         (2.02/sec)  (121.13/min)
Modifications:                649          (0.44/sec)  (26.53/min)
Adds:                         785          (0.53/sec)  (32.09/min)
Deletes:                      10           (0.01/sec)  (0.41/min)
Mod RDNs:                     6            (0.00/sec)  (0.25/min)
Compares:                     0            (0.00/sec)  (0.00/min)
Binds:                        324          (0.22/sec)  (13.25/min)

Proxied Auth Operations:      0
Persistent Searches:          17
Internal Operations:          0
Entry Operations:             0
Extended Operations:          4
Abandoned Requests:           0
Smart Referrals Received:     0

VLV Operations:               30
VLV Unindexed Searches:       0
VLV Unindexed Components:     20
SORT Operations:              22

Entire Search Base Queries:   12
Paged Searches:               2
Unindexed Searches:           0
Unindexed Components:         149

FDs Taken:                    249
FDs Returned:                 212
Highest FD Taken:             107

Broken Pipes:                 0
Connections Reset By Peer:    0
Resource Unavailable:         0
Max BER Size Exceeded:        0

Binds:                        324
Unbinds:                      155
---------------------------------
 - LDAP v2 Binds:             41
 - LDAP v3 Binds:             180
 - AUTOBINDs(LDAPI):          103
 - SSL Client Binds:          0
 - Failed SSL Client Binds:   0
```

```
 - SASL Binds:              134
   - EXTERNAL: 114
   - GSSAPI: 20
 - Directory Manager Binds:   10
 - Anonymous Binds:           1

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.

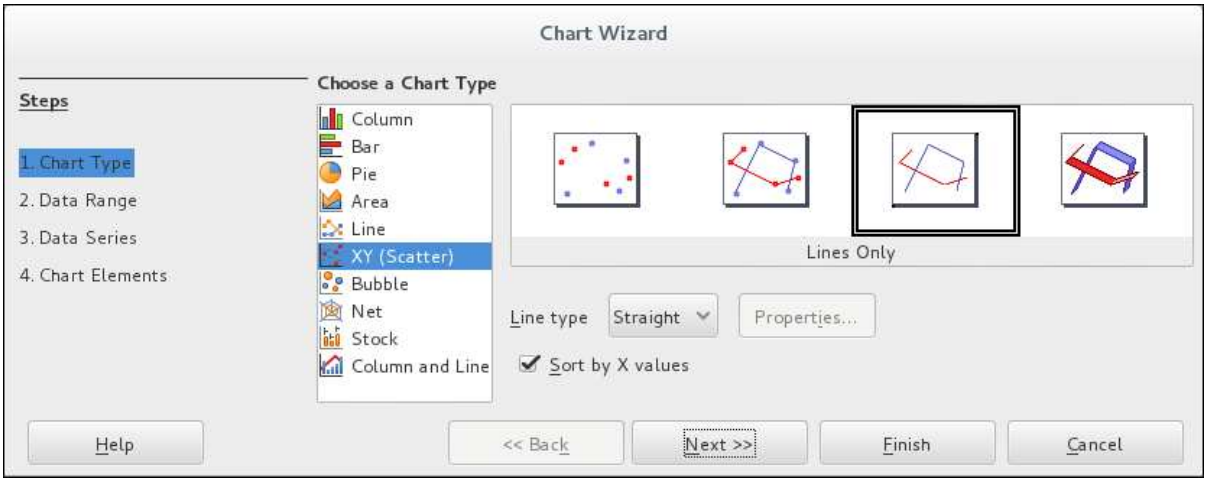
Each row in the file represents one unit of time, either minute or second, with total counts for that time period. The CSV file (for both per-minute and per-second statistics) contains the following columns, in order:

```
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 LibreOffice 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 LibreOffice 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.

20.5. 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 20.1, “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:

   ```
   # 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 ~]# systemctl restart dirsrv.target
   ```

Table 20.1. Disk Monitoring Configuration Attributes

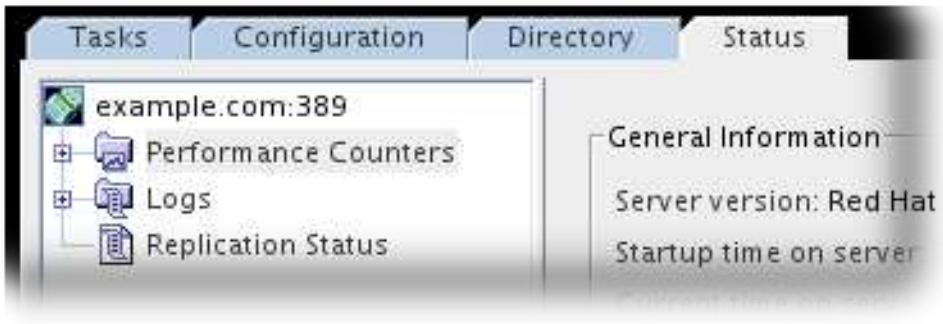
Configuration Attribute	Description
nsslapd-disk-monitoring	Enabled disk monitoring. This is the only required attribute, since the other configuration options have usable defaults.
nsslapd-disk-monitoring-grace-period	Sets a grace period to wait before shutting down the server after it hits half of 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).

20.6. 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.

20.6.1. 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.

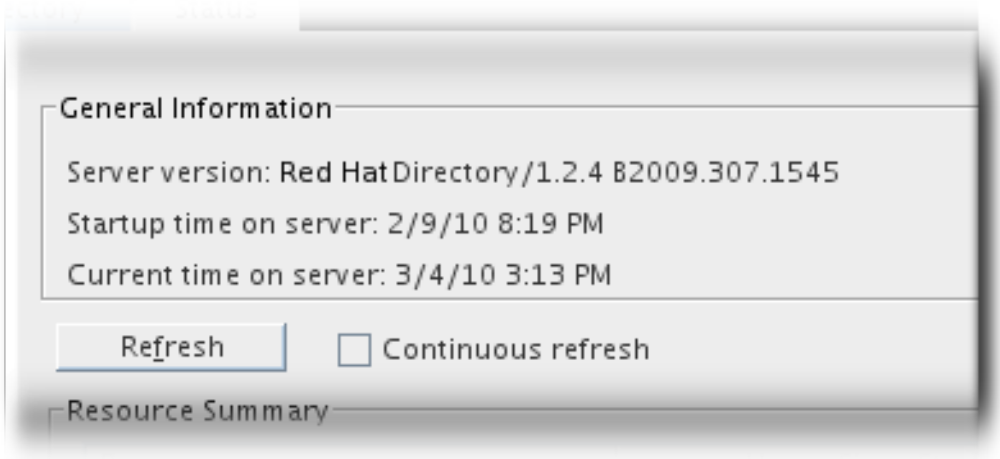


Table 20.2. General Information (Server)

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.

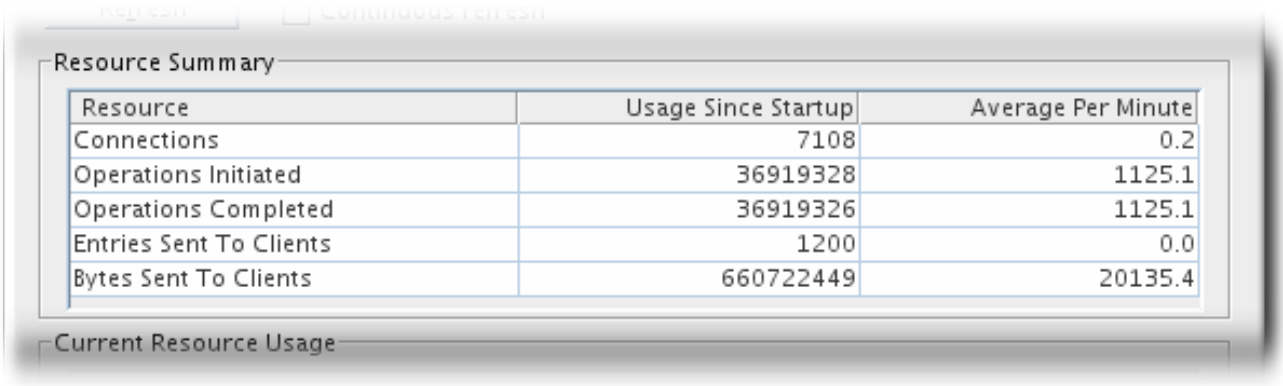


Table 20.3. Resource Summary

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.

Resource	Usage Since Startup	Average Per Minute
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.
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.

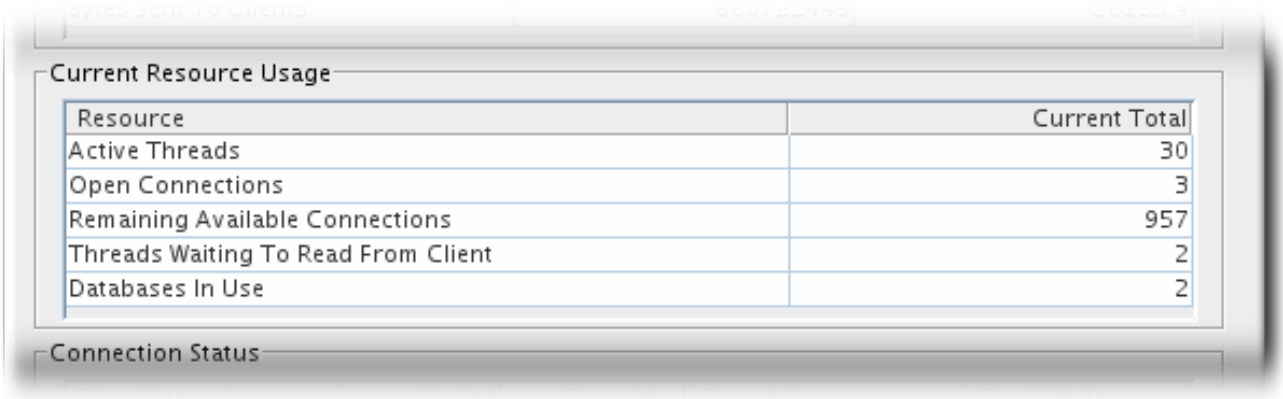


Table 20.4. Current Resource Usage

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.

Resource	Current Total
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.

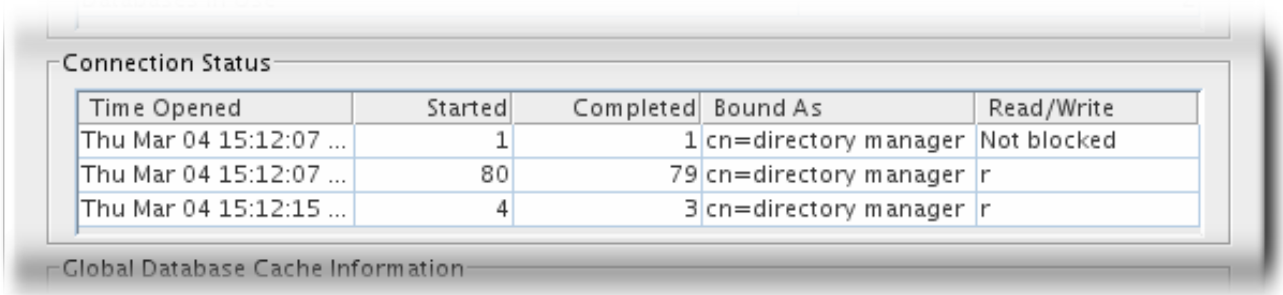
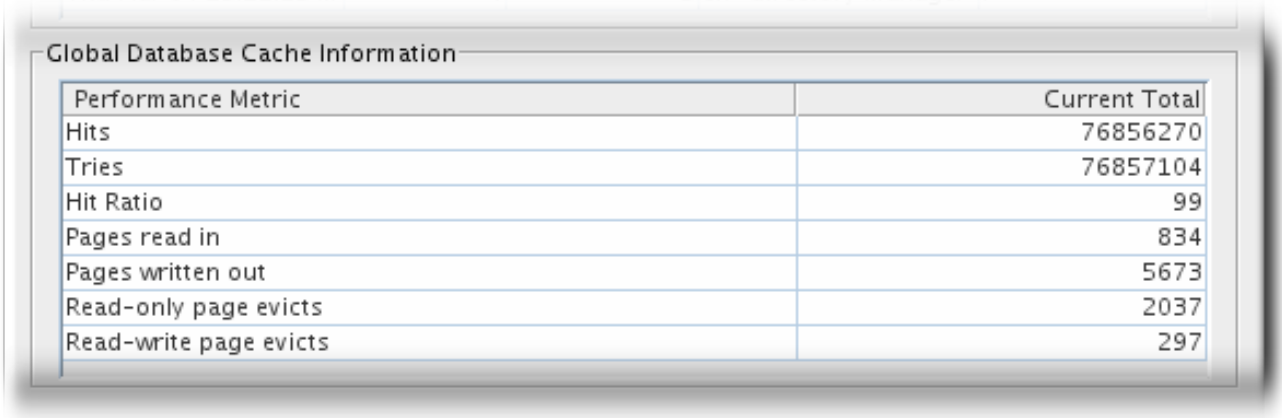


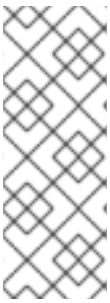
Table 20.5. Connection Status

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.

Table Header	Description
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.





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 20.7.2, “Monitoring Databases from the Command Line”.

Table 20.6. Global Database Cache Information

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.

20.6.2. 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 -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 14.3, “Using ldapsearch”.

Table 20.7. 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.
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.

Attribute	Description
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.

20.7. MONITORING DATABASE ACTIVITY

The database's current activities can be monitored through Directory Server Console or from the command line.

NOTE

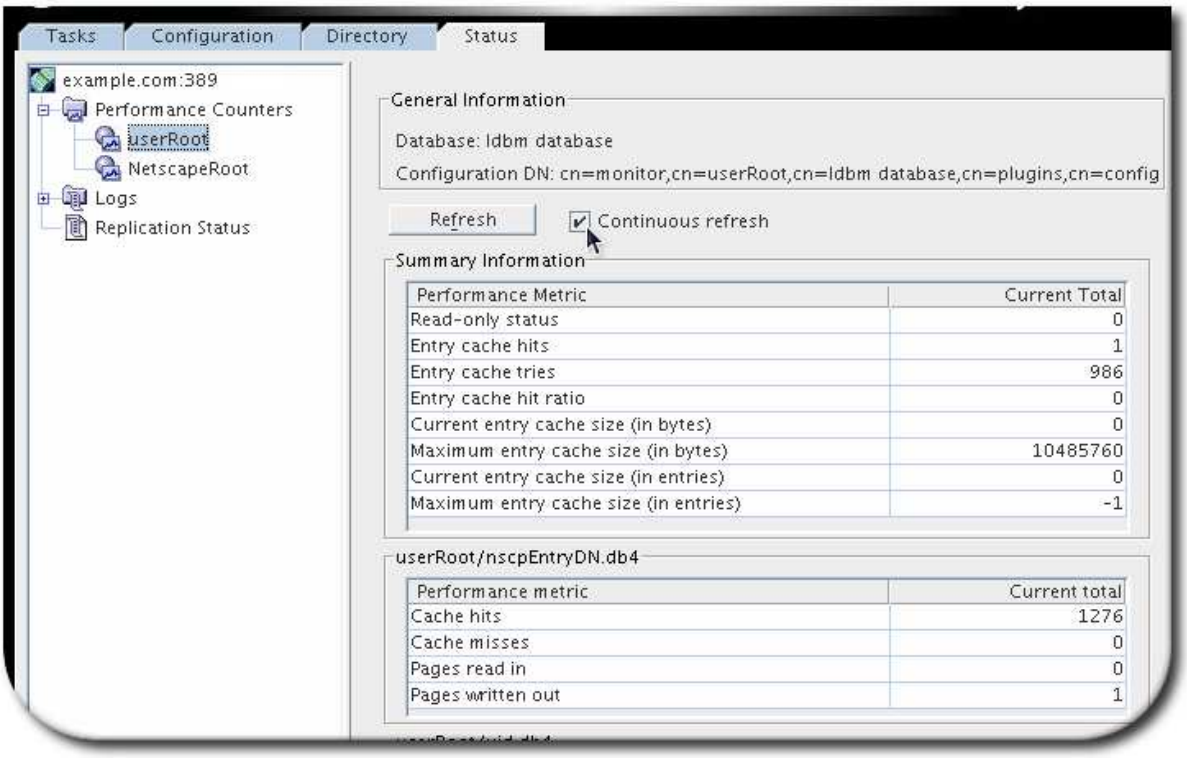
Tips for tuning the entry and database caches to improve server performance are in the *Red Hat Directory Server Performance Tuning Guide*.

20.7.1. 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 20.8. General Information (Database)

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 20.9. 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.

Performance Metric	Current Total
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 the ratio, enable the entry cache auto-tuning. For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.
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. The size of the entry cache is set in the ***nsslapd-cachememsize*** attribute in the **cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** entry. For optimized performance, enable entry cache auto-tuning. For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.
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.

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 20.10. 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 the ratio, enable the database cache auto-tuning. For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.
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 20.11. 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.

Performance Metric	Current Total
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.

20.7.2. 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 -p 389 -h server.example.com -x -s base -b
"cn=monitor,cn=database_name,cn=ldbm database,cn=plugins,cn=config" "(objectclass=*)"
```

Table 20.12. 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.

Attribute	Description
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 the ratio, enable the entry cache auto-tuning. For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.
currententrycachesize	The total size, in bytes, of directory entries currently present in the entry cache.
maxentrycachesize	The maximum size, in bytes, of directory entries that can be maintained in the entry cache. The size of the entry cache is set in the ***nsslapd-cachememsize*** attribute in the **cn=*database_name*,cn=ldbm database,cn=plugins,cn=config** entry. For optimized performance, enable entry cache auto-tuning. For details, see the corresponding section in the *Red Hat Directory Server Performance Tuning Guide*.
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.

Attribute	Description
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.

20.8. 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 -p 389 -h server.example.com -x -s sub -b
"cn=monitor,cn=DBLink1,cn=chaining database,cn=plugins,cn=config" "(objectclass=*)" nsAddCount
```

Table 20.13, “Database Link Monitoring Attributes” lists the database link monitoring attributes which can be monitored.

Table 20.13. Database Link Monitoring Attributes

Attribute Name	Description
nsAddCount	The number of add operations received.

Attribute Name	Description
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.

20.9. 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
```

CHAPTER 21. MONITORING DIRECTORY SERVER USING SNMP

The server and database activity monitoring log setup described in Chapter 20, *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.

21.1. 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.

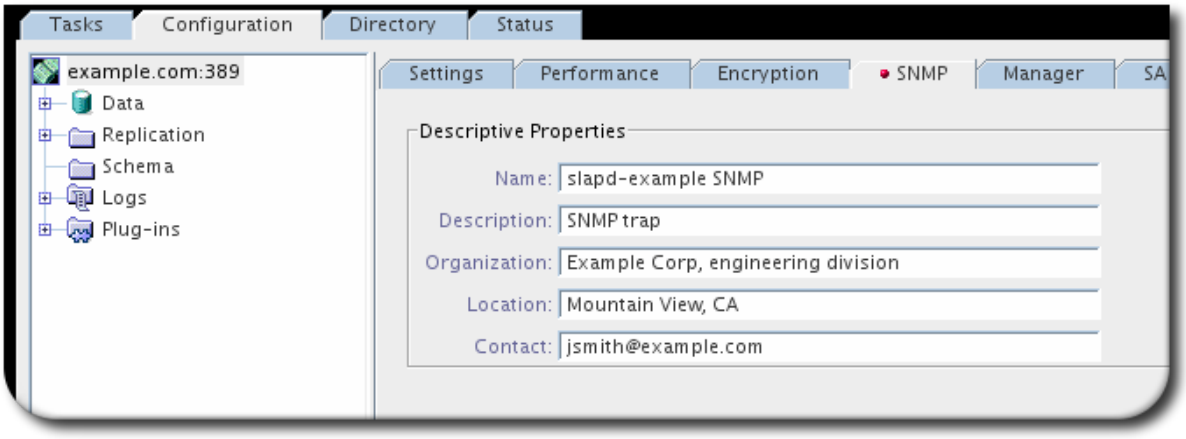
Values for SNMP attributes, otherwise known as variables, that can be queried are kept on the managed device and reported to the NMS as necessary. Each variable is known as a *managed object*, which is anything the agent can access and send to the NMS. All managed objects are defined in a management 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.

21.2. 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**.

21.3. SETTING UP AN SNMP AGENT FOR DIRECTORY SERVER

To query information from Directory Server using the SNMP protocol, set up an SNMP agent:

1. Install the 389-ds-base-snmp and net-snmp packages:

```
# yum install 389-ds-base-snmp net-snmp
```

2. To configure the SNMP master agent, edit the **/etc/snmp/snmpd.conf** file, adding the following entry to enable the agent extensibility (AgentX) protocol:

```
master agentx
```

   For further details about the AgentX protocol, see RFC 2741.
3. To configure the SNMP subagent, edit the **/etc/dirsrv/config/ldap-agent.conf** file, adding a server parameter for each Directory Server instance you want to monitor. For example:

```
server slapd-instance_name
```

4. Optionally, create an SNMP user account:

a. Stop the **snmpd** service:

```
# systemctl stop snmpd
```

b. Create the SNMP user account. For example:

```
# net-snmp-create-v3-user -A authentication_password -a SHA \
    -X private_password -x AES user_name
```

For details about the parameters used in the command, see the net-snmp-create-v3-user(1) man page.

c. Start the **snmpd** service:

```
# systemctl start snmpd
```

5. Optionally, set the Directory Server descriptive properties. For details, see Section 21.2, "Configuring the Directory Server for SNMP".

6. Start the **dirsrv-snmp** service:

```
# systemctl start dirsrv-snmp
```

7. Optionally, to verify the configuration:

a. Install the net-snmp-utils package:

```
# yum install net-snmp-utils
```

b. Query the Directory Server Object Identifiers (OID). For example:

```
# snmpwalk -v3 -u user_name -M /usr/share/snmp/mibs:/usr/share/dirsrv/mibs/ \
    -l AuthPriv -m +RHDS-MIB -A authentication_password -a SHA
    -X private_password -x AES server.example.com .1.3.6.1.4.1.2312.6.1.1
```

For further details about SNMP, see the *Monitoring Performance with Net-SNMP* section in the *Red Hat System Administrator's Guide*.

21.4. 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 21.5.3, "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.

21.5. 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 21.3, “Setting up an SNMP Agent for Directory Server”.

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 21.5.1, “Operations Table”
- Section 21.5.2, “Entries Table”
- Section 21.5.3, “Entity Table”
- Section 21.5.4, “Interaction Table”

NOTE

All of the Directory Server attributes monitored by SNMP use 64-bit integers for the counters, even on 32-bit systems.

21.5.1. Operations Table

The **Operations Table** provides statistical information about Directory Server access, operations, and errors. Table 21.1, “Operations Table: Managed Objects and Descriptions” describes the managed objects stored in the **Operations Table** of the **redhat-directory.mib** file.

Table 21.1. 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.
dsStrongAuthBinds	The number of binds to the directory that were established using a strong authentication method (such as TLS 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.

Managed Object	Description
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.

21.5.2. Entries Table

The **Entries Table** provides information about the contents of the directory entries. Table 21.2, "Entries Table: Managed Objects and Descriptions" describes the managed objects stored in the **Entries Table** in the **redhat-directory.mib** file.

Table 21.2. 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**.

21.5.3. 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 21.2, "Configuring the Directory Server for SNMP".

Table 21.3, "Entity Table: Managed Objects and Descriptions" describes the managed objects stored in the **Entity Table** of the **redhat-directory.mib** file.

Table 21.3. 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.

21.5.4. 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 21.4, "Interaction Table: Managed Objects and Descriptions" describes the managed objects stored in the **Interaction Table** of the **redhat-directory.mib** file.

Table 21.4. Interaction Table: Managed Objects and Descriptions

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.

Managed Object	Description
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 22. MAKING A HIGH-AVAILABILITY AND DISASTER RECOVERY PLAN

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.

22.1. 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 *Red Hat 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 22.1, "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 22.1. 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.

22.2. 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.

22.3. 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:

- Backing up databases and verifying the backups regularly
- Multi-master replication, chaining, backing up databases, and monitoring the server with a named pipe script
- Chaining

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.

22.3.1. 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 cron jobs. For example:

```
0 7 * * 1 /usr/sbin/db2bak.pl -Z instance_name
```

The **db2bak.pl** Perl script backs up the directory data without having to stop the server first.

NOTE

Red Hat recommends to back up the data on all servers in a multi-master replication environment.

Backing up both directory databases and the directory configuration (**dse.ldif** file) are covered in Section 6.3, “Backing up and Restoring Data”.

22.3.2. Multi-Master Replication for High-availability

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, frequently back up the database.

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 22.1. 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 15, *Managing Replication*.

22.3.3. Chaining Databases for High-availability

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 22.2. Scenarios for Chaining

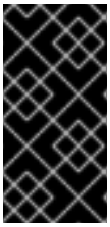
When chaining is combined with a list of failover servers, client traffic can be automatically redirected from a single server (or even group of servers) when they are offline. This does not help in recovery, but it helps manage the transition from primary to backup servers.

Chaining databases is covered in Section 2.3, “Creating and Maintaining Database Links” .

22.4. 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?



IMPORTANT

- Store a printed copy off the disaster recovery plan off-site.
- Test the disaster recovery plan on a regular basis and after configuration and infrastructure changes.

22.5. 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 uses the LDAP tools (such as **ldapsearch** and **ldapmodify**) supplied with OpenLDAP. The OpenLDAP tool options are described in the OpenLDAP man pages 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 14, *Finding Directory Entries*. More examples for using **ldapmodify** and **ldapdelete** are given in Section 3.1, "Managing Entries Using the Command Line".

A.1. 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.2. 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 -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 -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 -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 -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.3. CHANGING PASSWORDS

The **ldappasswd** command can either set a new user-defined password or generate a new password for an account. Table 19.1, "ldappasswd Options" 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 -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 TLS, Start TLS, or SASL. For information on how to configure TLS for LDAP clients, see Section 9.8.4, "Authenticating Using a Certificate" .

For a list of password operation-related parameters for **ldapasswd**, see Table 19.1, "ldappasswd Options".

Example A.3. Directory Manager Changing a User's Password Over TLS

The Directory Manager changes the password of the user **uid=tuser1,ou=People,dc=example,dc=com** to new_password over TLS.

```
# ldappasswd -D "cn=Directory Manager" -W -ZZ -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 TLS.

```
# ldappasswd -D "cn=Directory Manager" -W -ZZ -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 TLS.

```
# ldappasswd -p 389 -h server.example.com -ZZ -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
```

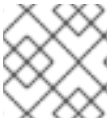
Example A.7. User Already Authenticated by Kerberos Prompts for a New Password

A user, who has already authenticated by Kerberos, prompts for the new password. This is not performed over TLS.

```
# ldappasswd -p 389 -h server.example.com -O noplain,minssf=1,maxbufsize=512 -I
```

A.4. 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.1, "ldapurl Parameters"; the full list of parameters are in the OpenLDAP manpages.

Table A.1. ldapurl Parameters

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, *Managing 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.

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.

Field	Definition
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 10 Configuration, Command, and File Reference* for a list of standard object classes and Chapter 12, *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 10 Configuration, Command, and File Reference* for a list of standard attributes and Chapter 12, *Managing the Directory Schema* for information on customizing the schema.
[*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 3.2.3.5, "Adding an Attribute Subtype". For a complete list of the supported subtypes tags, see Table D.1, "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, *Managing 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 TLS 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 10 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 10 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 10 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**.
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 10 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 10 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.
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 10 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.

LDIF Element	Description
list_of_attributes	Specifies the list of optional attributes to maintain for the entry. See the *Red Hat Directory Server 10 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 *Red Hat Directory Server Configuration, Command, and File 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 6.1.2, "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 6.1.4.1, "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 3.1.3, "Adding an Entry" .

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
```

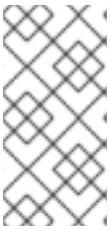
B.6. STORING INFORMATION IN MULTIPLE LANGUAGES

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.

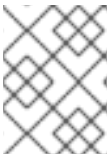
C.1. COMPONENTS OF AN LDAP URL

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 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 **192.0.2.90**.
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.

Component	Description
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.
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

Unsafe Character	Escape Characters
>	%3e
"	%22
#	%23
%	%25
{	%7b
}	%7d
|	%7c
\	%5c
^	%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)
```

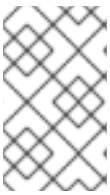
- Because no attributes are specified, the search returns all attributes.
- Because the search scope is **sub**, the search encompasses the base entry **dc=example,dc=com** and entries at all levels under the base entry.

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 Czech 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 13, *Managing Indexes*.

For a list of language tags and OIDs supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

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 3.2.3.5, "Adding an Attribute Subtype" . Table D.1, "Supported Language Subtypes" lists the supported language subtypes for Directory Server.

Table D.1. Supported Language Subtypes

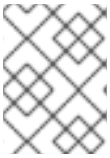
Language Tag	Language
af	Afrikaans
be	Belorussian
bg	Bulgarian
ca	Catalan
cs	Czech
da	Danish
de	German
el	Greek
en	English
es	Spanish

Language Tag	Language
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	Slovak
sl	Slovenian
sq	Albanian
sr	Serbian

Language Tag	Language
sv	Swedish
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 14.4, "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 OIDs supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

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 language tags supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file.

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
```

NOTE

This syntax is only supported by the **mozldap** utility and not by OpenLDAP utilities, such as **ldapsearch**.

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 OIDs supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file. For a list of relational operators and their equivalent suffixes, see Table D.2, "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
```

NOTE

This syntax is only supported by the **mozldap** utility and not by OpenLDAP utilities, such as **ldapsearch**.

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 language tags supported by the Directory Server, see the **/etc/dirsrv/config/slapd-collations.conf** file. For a list of relational operators and their equivalent suffixes, see Table D.2, "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.2, "Search Types, Operators, and Suffixes" summarizes each type of search, the operator, and the equivalent suffix.

Table D.2. 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 Czech 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 -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 -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 -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 -b "dc=example,dc=com"
"sn:de.3:=passin"
```

APPENDIX E. MANAGING THE ADMINISTRATION SERVER

E.1. INTRODUCTION TO RED HAT ADMINISTRATION 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

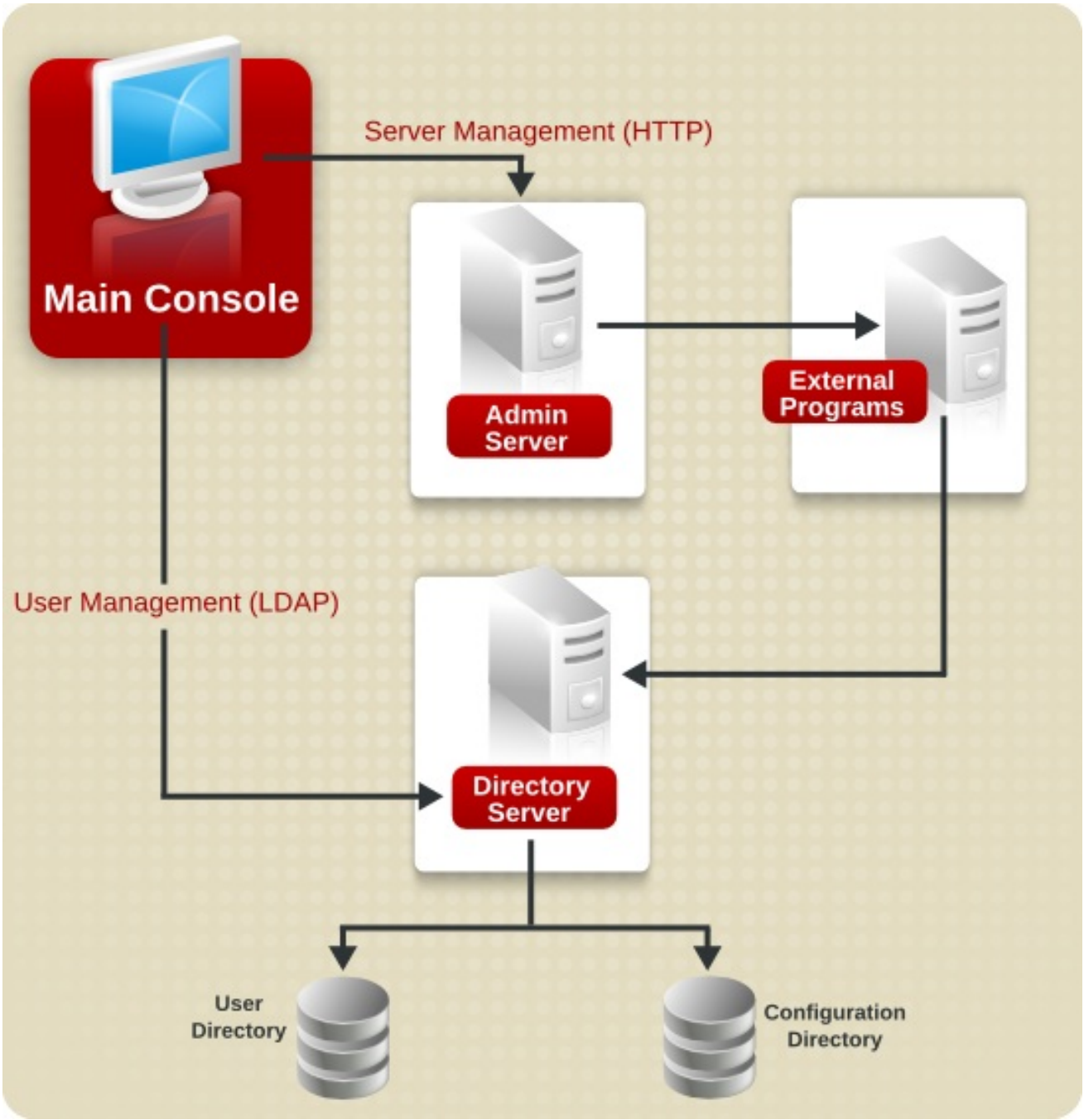


Figure E.1. Interactions between the Console, Administration Server and Directory Server

The Administration 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 Administration 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 Administration Server processes server configuration requests for Red Hat Directory Server, the Configuration Directory Server and User Directory Server instances are both defined in the Administration Server configuration.

As a web server, the Administration 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 Administration Server both over secure and standard connections, since the Administration Server supports both HTTP or HTTPS, if TLS is enabled.

When Red Hat Directory Server or Red Hat Certificate System (which depends on Red Hat Directory Server) is installed, then the Administration 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 Administration Server.

There can be *only one* Administration Server per machine. This single Administration Server instance can handle multiple instances of Directory Server and other clients which can use the Administration 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 Administration Server instance to perform the requested tasks. For example, Administration 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 Administration Server itself can be managed through its own Java-based interface, by editing its configuration files, or through command-line tools.

E.2. ADMINISTRATION SERVER CONFIGURATION

The Administration Server is a separate server from Red Hat Directory Server or Red Hat Certificate System, although they work interdependently. The Administration Server processes, file locations, and configuration options are also separate. This chapter covers the Administration Server information, including starting and stopping the Administration Server, enabling TLS, viewing logs, and changing Administration Server configuration properties, such as the server port number.

E.2.1. File Locations

See the corresponding section in the *Red Hat Directory Server Configuration, Command, and File Reference*.

E.2.2. Opening the Administration 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 Administration Server prompts for the user name, password, and Administration Server location. The Administration Server location is a URL; for a standard connection, this has the **http:** prefix for a standard HTTP protocol. If TLS is enabled, then this uses the **https:** prefix

for the secure HTTPS protocol.

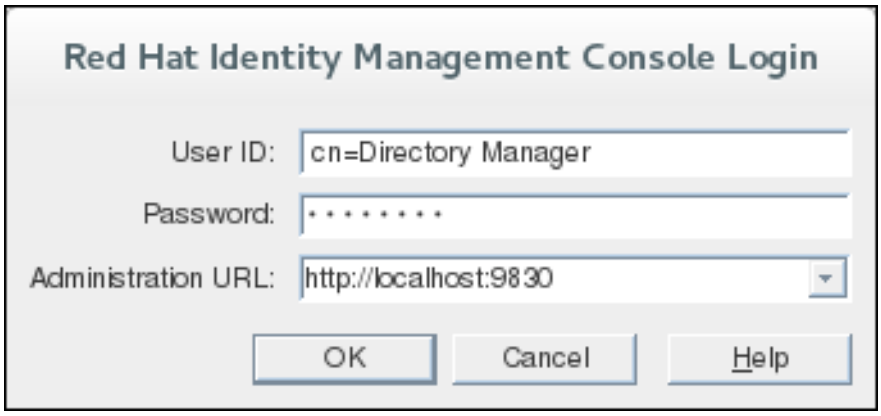
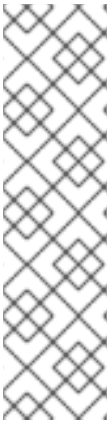


Figure E.2. Login Box



NOTE

It is possible to send the Administration 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 Administration 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 Administration Server Console, select the Administration Server instance from the server group on the left, and then click the **Open** at the top right of the window.

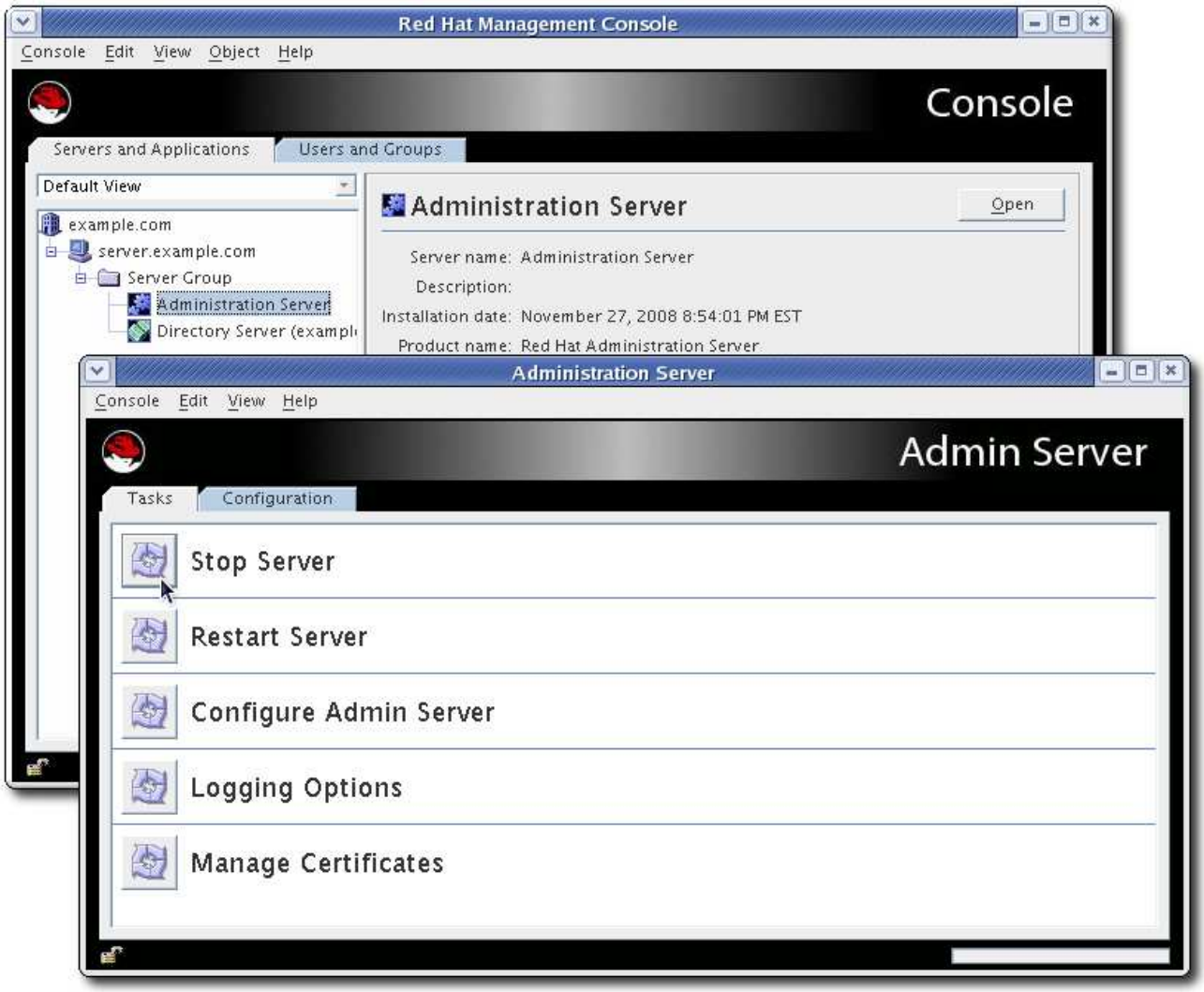


Figure E.3. The Administration Server Console

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.2.3. Viewing Logs

Log files monitor activity for Administration Server and can help troubleshoot server problems. Administration Server logs use the Common Logfile Format, a broadly supported format that provides information about the server.

Administration Server generates two kinds of logs:

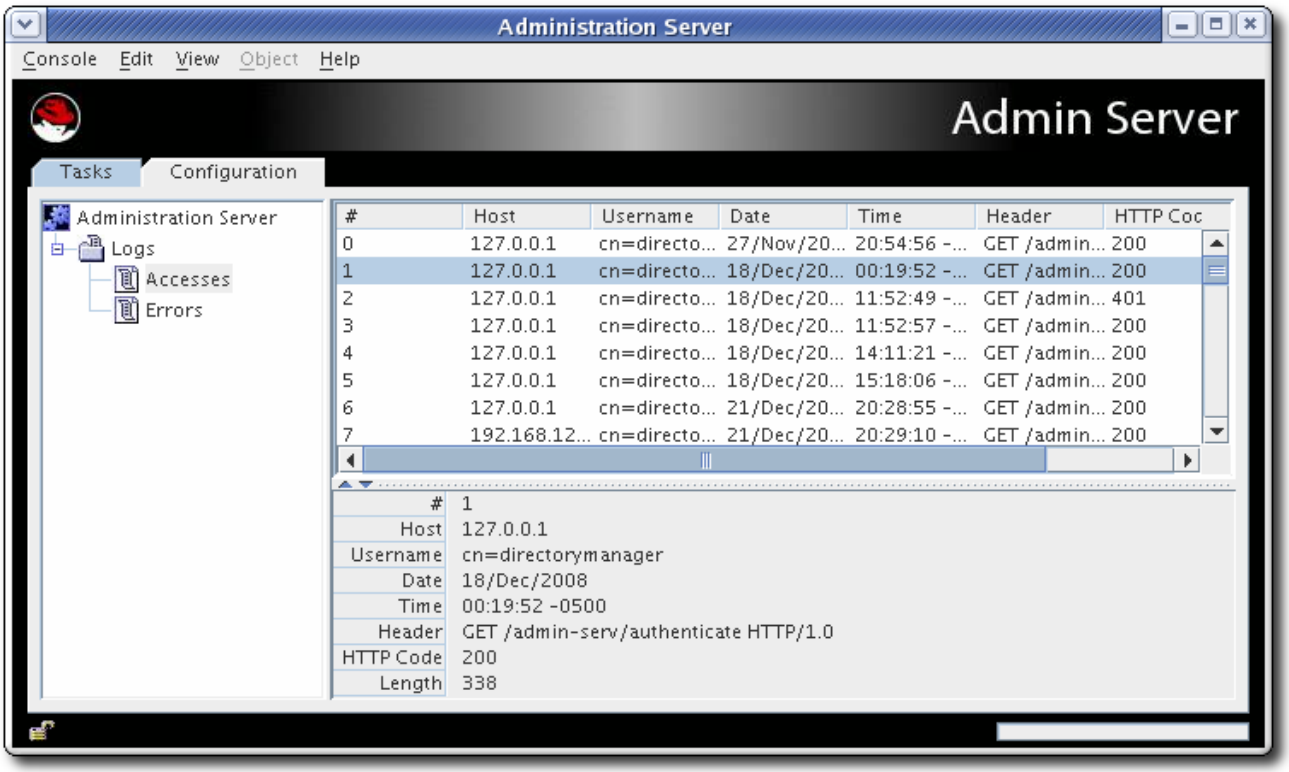
- *Access logs*. Access logs show requests to and responses from the Administration Server. By default, the file is located at **/var/log/dirsrv/admin-serv/access**.
- *Error logs*. Error logs show messages for errors which the server has encountered since the log file was created. It also contains informational 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-serv/error**.

The logs can be viewed through Administration Server Console or by opening the log file.

E.2.3.1. Viewing the Logs through the Console

1. Open the Administration Server management window.
2. Click the **Configuration** tab.
3. Expand the **Logs** directory, and click the log file name, either **Accesses** or **Error**.



E.2.3.2. Viewing Logs in the Command Line

The access log, by default, is at **/var/log/dirsrv/admin-serv/error**. To view the access log, open it in an editor such as **vi**.

Access logs show connections to the Administration 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/2008:19:32:52.157345975 -0500] "GET /admin-serv/authenticate HTTP/1.0" 200 338
> 192.168.123.121 - cn=Directory Manager [23/Dec/2008:19:33:14.453724501 -0500] "POST /admin-serv/tasks/Configuration/ServerSetup HTTP/1.0" 200 244
> 192.168.123.121 - cn=Directory Manager [23/Dec/2008:19:33:16.573485244 -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-serv/errors**. To view the error log, open it in an editor such as **vi**.

Error logs record any problem response from the Administration 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".

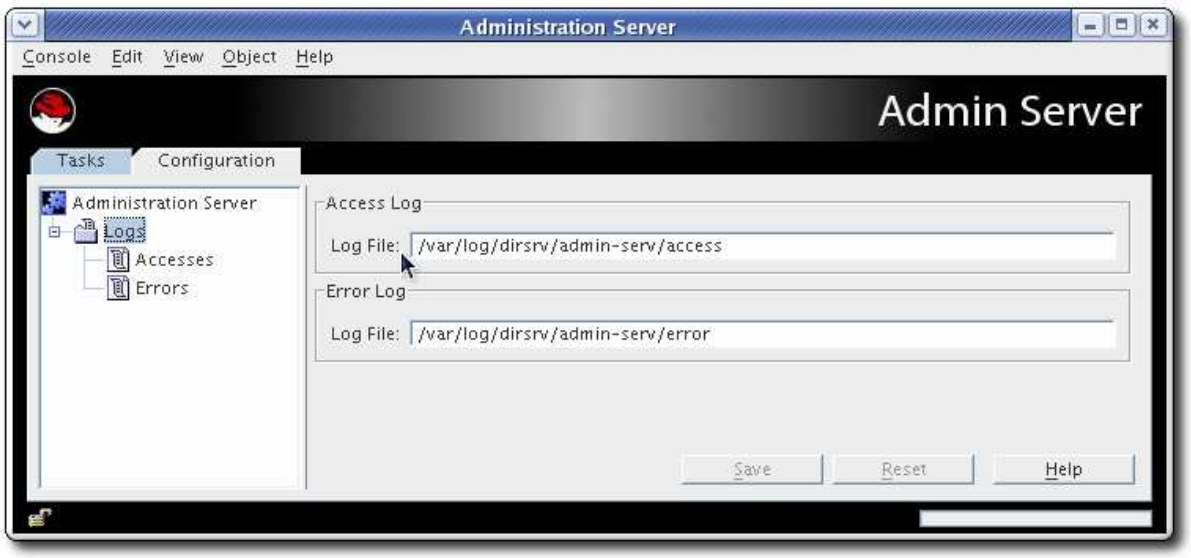
Example E.2. Example Error Logs

```
[24/Mar/2017:11:14:27.110314677 +0100] - NOTICE - ldbm_back_start - total cache size:
417775616 B;
[24/Mar/2017:11:14:27.165466639 +0100] - INFO - dblayer_start - Resizing db cache size:
1519206400 -> 132562944
[24/Mar/2017:11:14:27.650899322 +0100] - INFO - slapd_daemon - slapd started.  Listening on
All Interfaces port 389 for LDAP requests
[24/Mar/2017:11:14:29.620268885 +0100] - WARN - modify_config_dse - Modification of attribute
"aci" is not allowed, ignoring!
```

E.2.3.3. 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 Administration 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.2.3.4. 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 Administration 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 Administration Server configuration entry in the Configuration Directory Server.

    1. Get the name of the Administration Server entry. Since the Administration 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 -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 Administration 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 Administration 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 Administration Server.

```
# systemctl restart dirsrv-admin.service
```

E.2.3.5. 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 Administration Server. This is faster for the Administration Server, since it does not have to do a DNS lookup for every connection. It is possible to set the Administration 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 Administration Server to perform DNS lookups:

1. Edit the **console.conf** file for the Administration 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
```

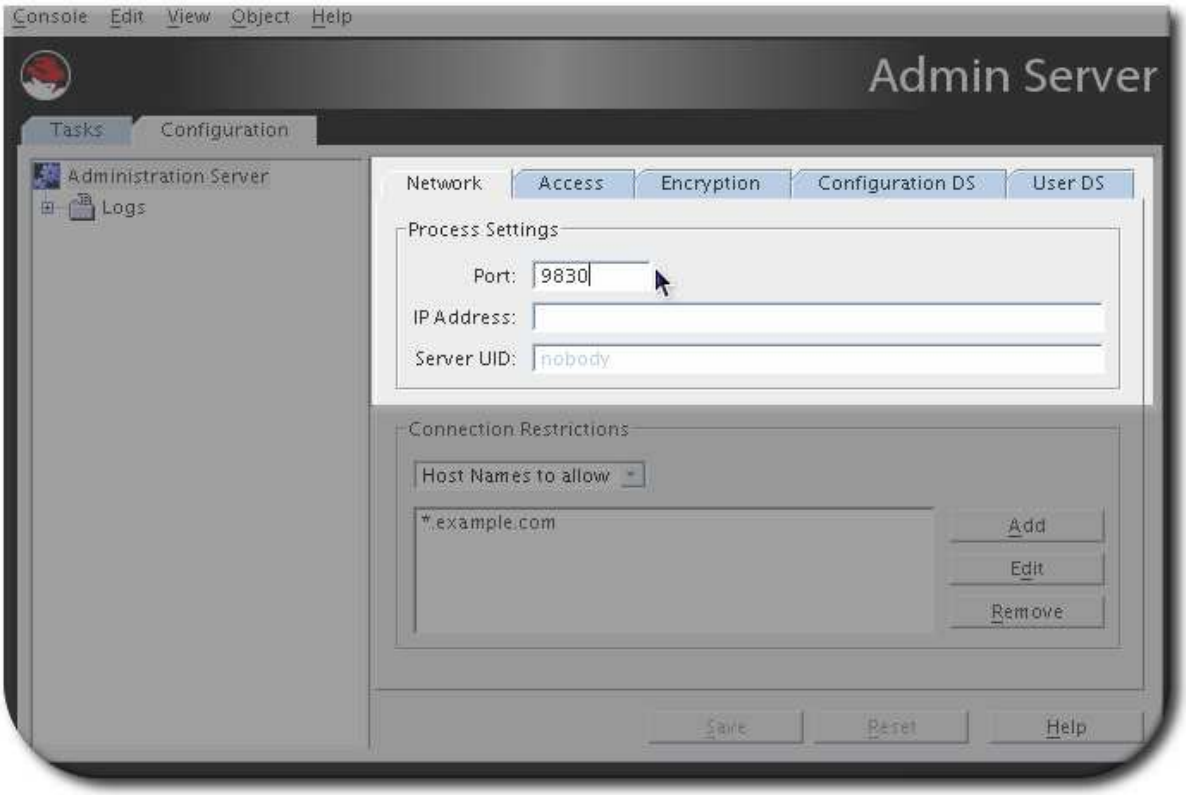
E.2.4. Changing the Port Number

The *port number* specifies where an instance of Administration Server listens for messages.

The default port number for Administration 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.2.4.1. Changing the Port Number in the Console

1. Open the Administration Server management window.
2. Click the **Configuration** tab.
3. Click the **Network** tab.



4. Enter the port number for the Administration Server instance in the **Port** field. The Administration 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 Administration Server port number in the connection URL.

E.2.4.2. Changing the Port Number in the Command Line

The port number for the Administration Server is **9830** by default.

The Administration 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 Administration Server configuration entry in the Configuration Directory Server.

    1. Get the name of the Administration Server entry. Since the Administration 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 -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 Administration 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 Administration 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 Administration Server.

    ```
    # systemctl restart dirsrv-admin.service
    ```

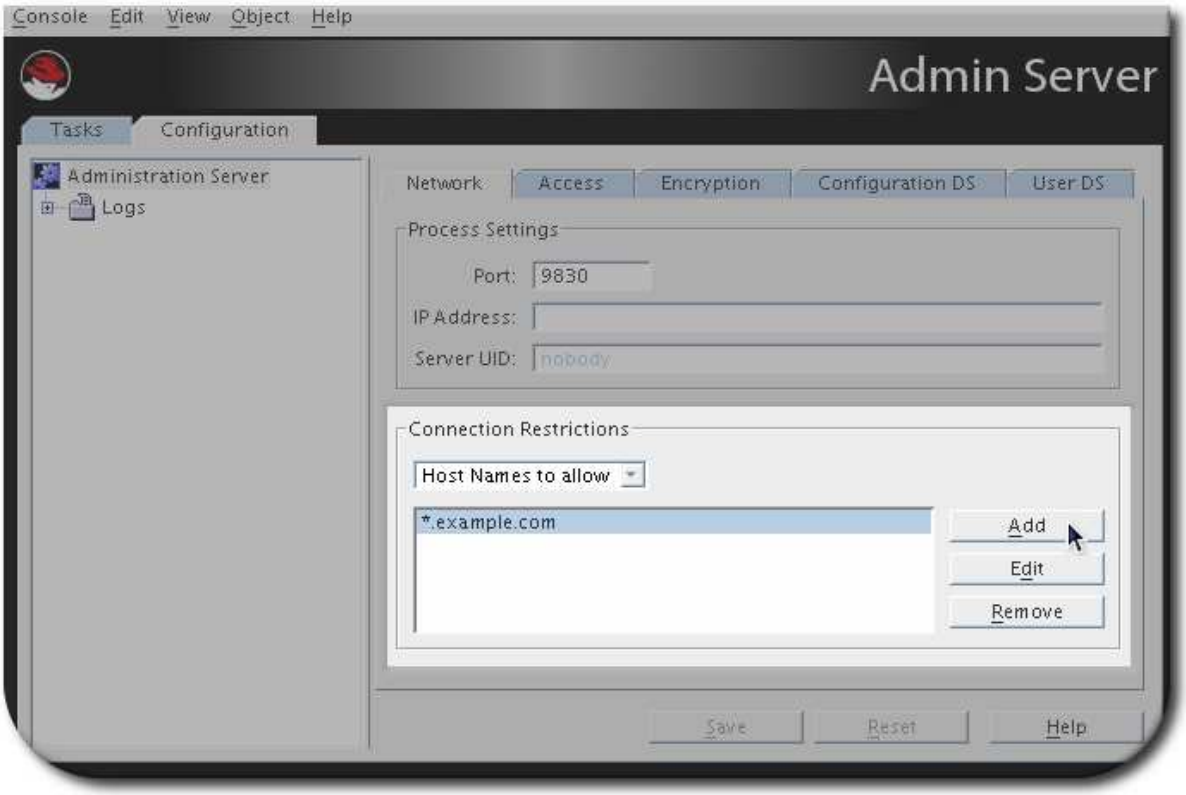
E.2.5. Setting Host Restrictions

Connection restrictions specify which hosts are allowed to connect to the Administration 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 Administration Server. This setting allows wildcards within a domain or an IP address range to make setting connection restrictions simpler.

E.2.5.1. Setting Host Restrictions in the Console

1. Open the Administration 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 Administration 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 name 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 Administration 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**. **Administration Server**.

E.2.5.2. Setting Host Restrictions in the Command Line

Host restrictions sets rules for what network clients can connect to the Administration Server and, therefore, to services which use the Administration Server. There are two kinds of host restrictions, restrictions based on the host or domain name and restrictions based on the IP address.

The Administration 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 Administration Server supports both IPv4 and IPv6 addresses.

The Administration Server entry can be edited using **ldapmodify**.

To set host restrictions:

1. Get the name of the Administration Server entry. Since the Administration 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 -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 Administration Server to apply the changes.

```
# systemctl restart dirsrv-admin.service
```

E.2.6. 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 *Red Hat Directory Server Administration 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 Administration Server. The information for the admin user is set on the **Access** tab in the Console.

NOTE

The Administration 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 Administration Server Console.

To change the Administration Server Administrator's ID or password:

1. Open the Administration 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 Administration Server.

5. Click **Save**.

E.2.7. Working with TLS

The Administration Server can run over HTTPS (secure HTTP) if TLS is enabled on the server. There are steps to enabling TLS:

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 Administration Server configuration to allow TLS connections.

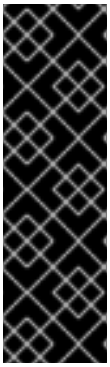
E.2.7.1. Managing Certificates for Administration Server

To request and install certificates for the Administration Server Console, follow the procedures for the Directory Server Console. See:

- Section 9.3.1, “Creating the NSS Database for a Directory Server Instance”
- Section 9.3.2, “Creating a Certificate Signing Request”

  To use the same certificate for Directory Server as for the Administration Server, see Section E.2.7.1.1, “Using the Directory Server Private Key and Certificate for the Admin Server” .
- Section 9.3.3, “Installing a CA Certificate”

  To use the same certificate for Directory Server as for the Administration Server, see Section E.2.7.1.1, “Using the Directory Server Private Key and Certificate for the Admin Server” .
- Section 9.3.4, “Installing a Certificate”
- Section 9.3.6, “Renewing a Certificate”
- Section 9.3.9, “Changing the CA Trust Options”
- Section 9.3.10, “Changing the Password of the NSS Database”
- Section 9.3.11, “Adding a Certificate Revocation List”



IMPORTANT

If you use:

- The graphical user interface, perform the steps in the **Manage Certificates** menu of the Administration Server Console instead of the Directory Server Console.
- The command line, use the **/etc/dirsrv/admin-serv/** instead of the **/etc/dirsrv/slapd-*instance_name*/** directory when you manage the Network Security Services (NSS) database.

E.2.7.1.1. Using the Directory Server Private Key and Certificate for the Admin Server

The Administration Server and Directory Server use different PKI databases. When the **Certificate Request Wizard** for the Directory Server is passed through, the automatically generated private key is stored in the Directory Server's PKI database. However, because the same private key does not exist in both databases, a certificate issued for one cannot be installed in the other database.

Run the following commands to export the private key and certificate of the Directory Server, and import them into the Administration Server's database:

1. Shut down the Administration Server:

   ```
   # systemctl stop dirsrv-admin
   ```

2. Shut down the Directory Server:

   ```
   # systemctl stop dirsrv@instance
   ```

3. List the contents of the Directory Server NSS database:

```
# certutil -L -d /etc/dirsrv/admin-serv/

Certificate Nickname                 Trust Attributes
                                     SSL,S/MIME,JAR/XPI

Demo CA                               CT,,
server-cert                          u,u,u
```

4. Export the private key and certificate with the name *server-cert* from the Directory Server's PKI database:

```
# pk12util -o /tmp/keys.pk12 -n server-cert -d /etc/dirsrv/slapd-instance/
Enter Password or Pin for "NSS Certificate DB":
Enter password for PKCS12 file:
Re-enter password:
pk12util: PKCS12 EXPORT SUCCESSFUL
```

   Enter the Directory Server's key store password, and optionally a new password for the temporarily exported file when prompted.

5. Import the private key and certificate into the Administration Server's PKI database:

```
# pk12util -i /tmp/keys.pk12 -d /etc/dirsrv/admin-serv/
Enter a password which will be used to encrypt your keys.
The password should be at least 8 characters long,
and should contain at least one non-alphabetic character.

Enter new password:
Re-enter password:
Enter password for PKCS12 file:
pk12util: PKCS12 IMPORT SUCCESSFUL
```

   pk12util asks you to set a password for the Administration Server's key store. If you already had set one to this database before, you are prompted to enter this password instead. If you set a password on the exported file in the previous step, you are additionally asked to enter this one as well.

6. Delete the temporarily exported file:

```
# rm /tmp/keys.pk12
```

7. Trust the *Demo CA*:

```
# certutil -M -d /etc/dirsrv/admin-serv/ -n "Demo CA" -t CT,,
```

8. Start the Directory Server:

```
# systemctl start dirsrv@instance
```

9. Start the Administration Server:

```
# systemctl start dirsrv-admin
```

E.2.7.2. Enabling TLS

See Section 9.4.3, "Enabling TLS in the Administration Server" .

E.2.7.3. Creating a Password File for the Administration Server

Normally, if TLS is enabled, the server prompts for a security password when the Administration Server is restarted:

```
Starting dirsrv-admin:
Please enter password for "internal" token:
```

The Administration Server can use a password file when TLS 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. Create the **/etc/dirsrv/admin-serv/password.conf** file with the following contents:
   - For a system with Federal Information Processing Standard (FIPS) mode disabled:

     ```
     internal:password
     ```

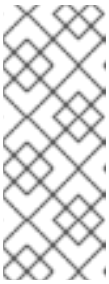
   - For a system with FIPS mode enabled:

     ```
     internal:password
     NSS FIPS 140-2 Certificate DB:password
     ```

   Lines in this file use the following format: ***token_name*:*password***.

   For the NSS software crypto module (the default software database), the token is always called **internal**. If FIPS mode is enabled, the additional token for the certificate database is always called **NSS FIPS 140-2 Certificate DB**.

2. The password file should be owned by the Administration Server user and set to read-only by the Administration Server user, with no access to any other user (mode **0400**).



NOTE

To find out what the Administration Server user ID is, run **grep** in the Administration Server configuration directory:

```
# grep "^User" /etc/dirsrv/admin-serv/console.conf
User dirsrv
```

To set the permissions, enter:

```
# chown dirsrv:root /etc/dirsrv/admin-serv/password.conf
# chmod 0400 /etc/dirsrv/admin-serv/password.conf
```

3. Edit the **/etc/dirsrv/admin-serv/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 Administration Server:

```
# systemctl restart dirsrv-admin.service
```

After TLS is enabled, then the Administration Server can only be connected to using HTTPS. All of the previous HTTP (standard) URLs for connecting to the Administration Server and its services no longer work. This is true whether connecting to the Administration Server using the Console or using a web browser.

E.2.8. Changing Directory Server Settings

The Administration 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 Administration Server configuration so that it communicates with a different Directory Server instance.

E.2.8.1. 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 Administration 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 TLS.

5. Click **Save**.

E.2.8.2. Changing the User Directory Host or Port

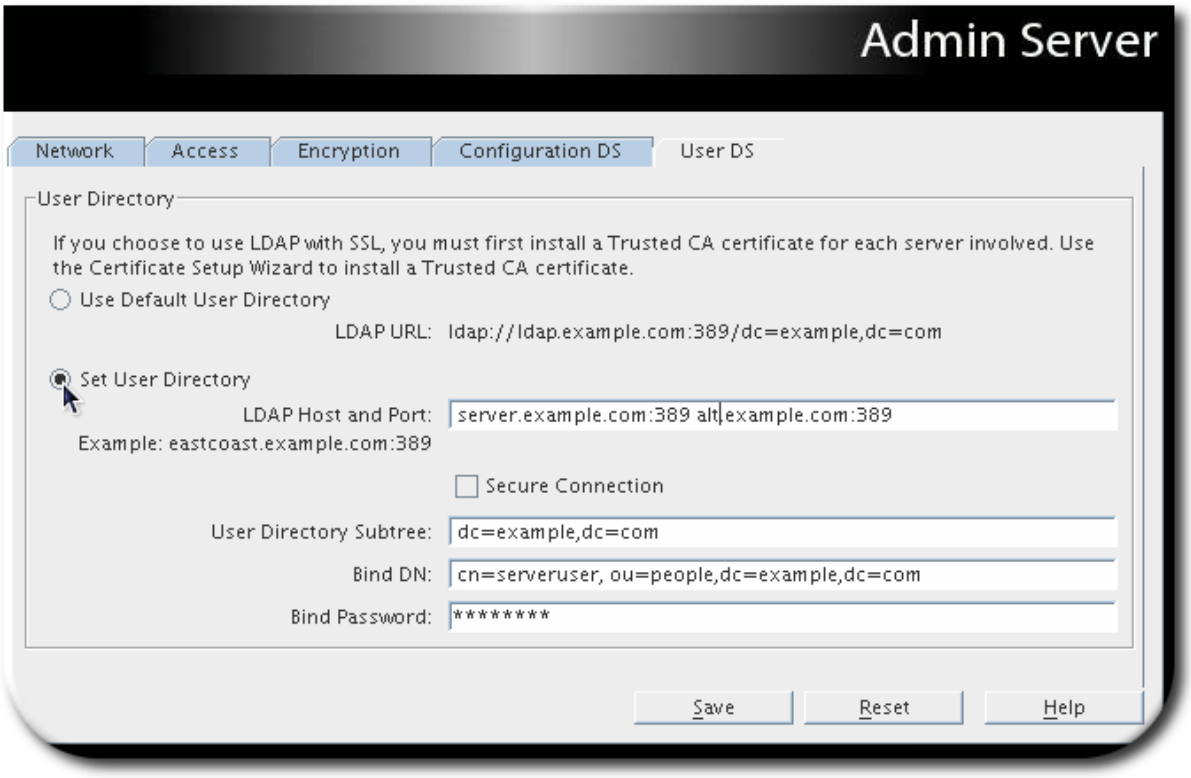
The user directory is used for authentication, user management, and access control. It stores all user and group data, account data, group lists, and access control instructions (ACIs).

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.

Administration Server can be configured to authenticate users against multiple user directories.

To change the information for the user directory:

1. Open the Administration 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 TLS to connect to the user directory. *Only* select this if the Directory Server is already configured to use TLS.

- 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 Administration Server services pages URL is the Administration 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 TLS is enabled on the Administration 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.

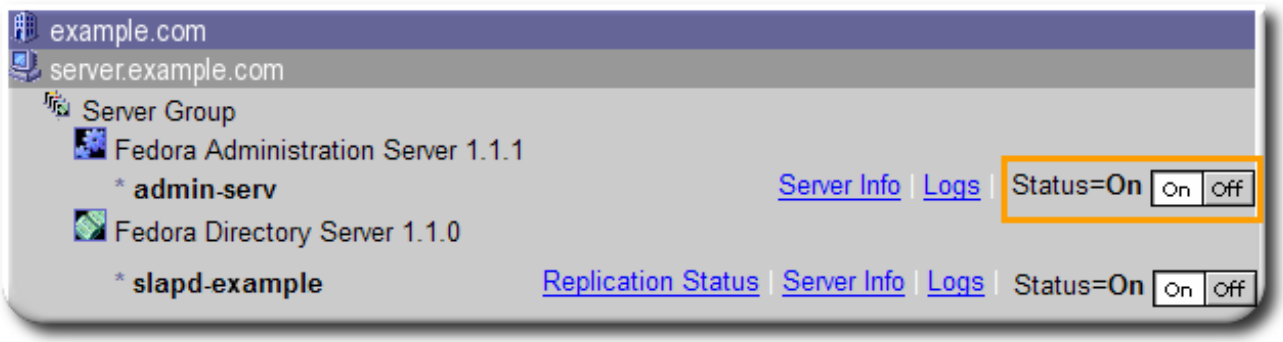
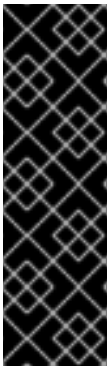


Figure F.1. Stopping and Stopping Servers



IMPORTANT

If either the Administration 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 Administration 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 Administration 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**.

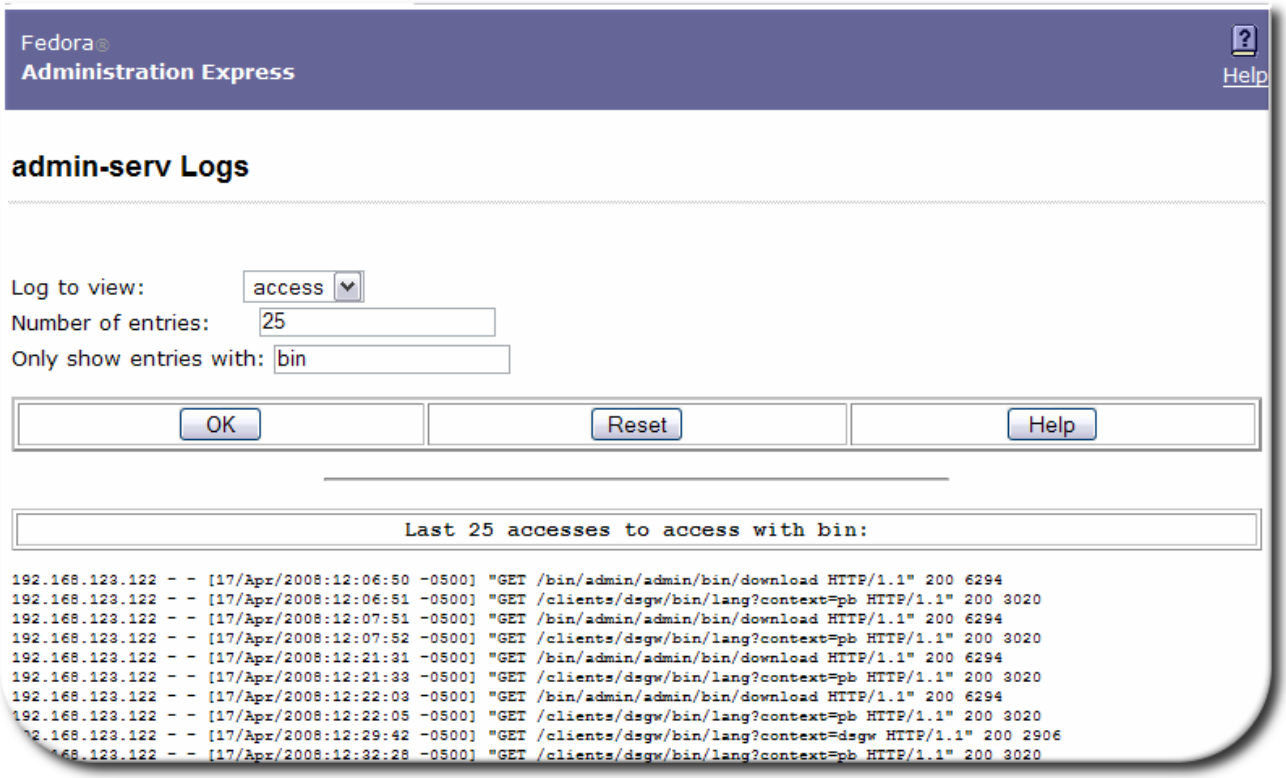


Figure F.2. Checking Logs

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.

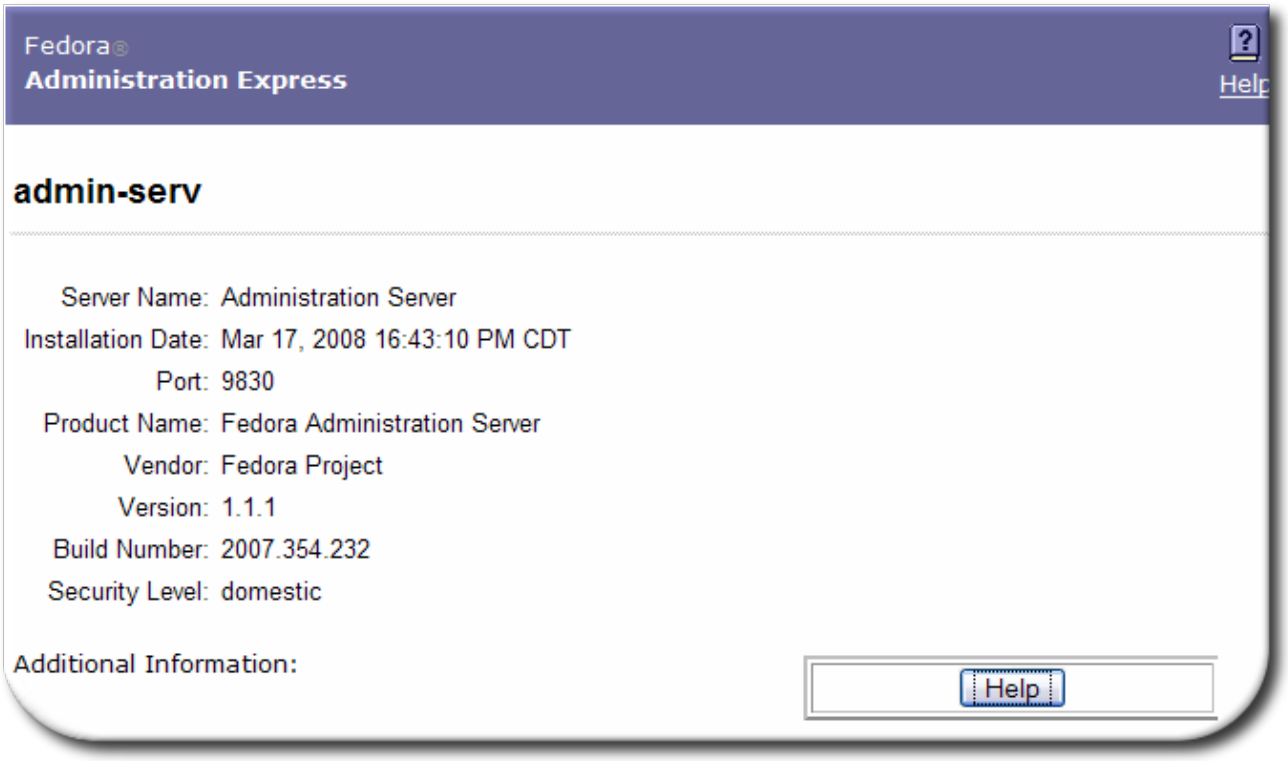


Figure F.3. Checking Server Information

The Directory Server information is located in the **/etc/dirsrv/slapd-*instance*/dse.ldif** file; the Administration 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 Administration 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 Administration 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 Administration 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**.

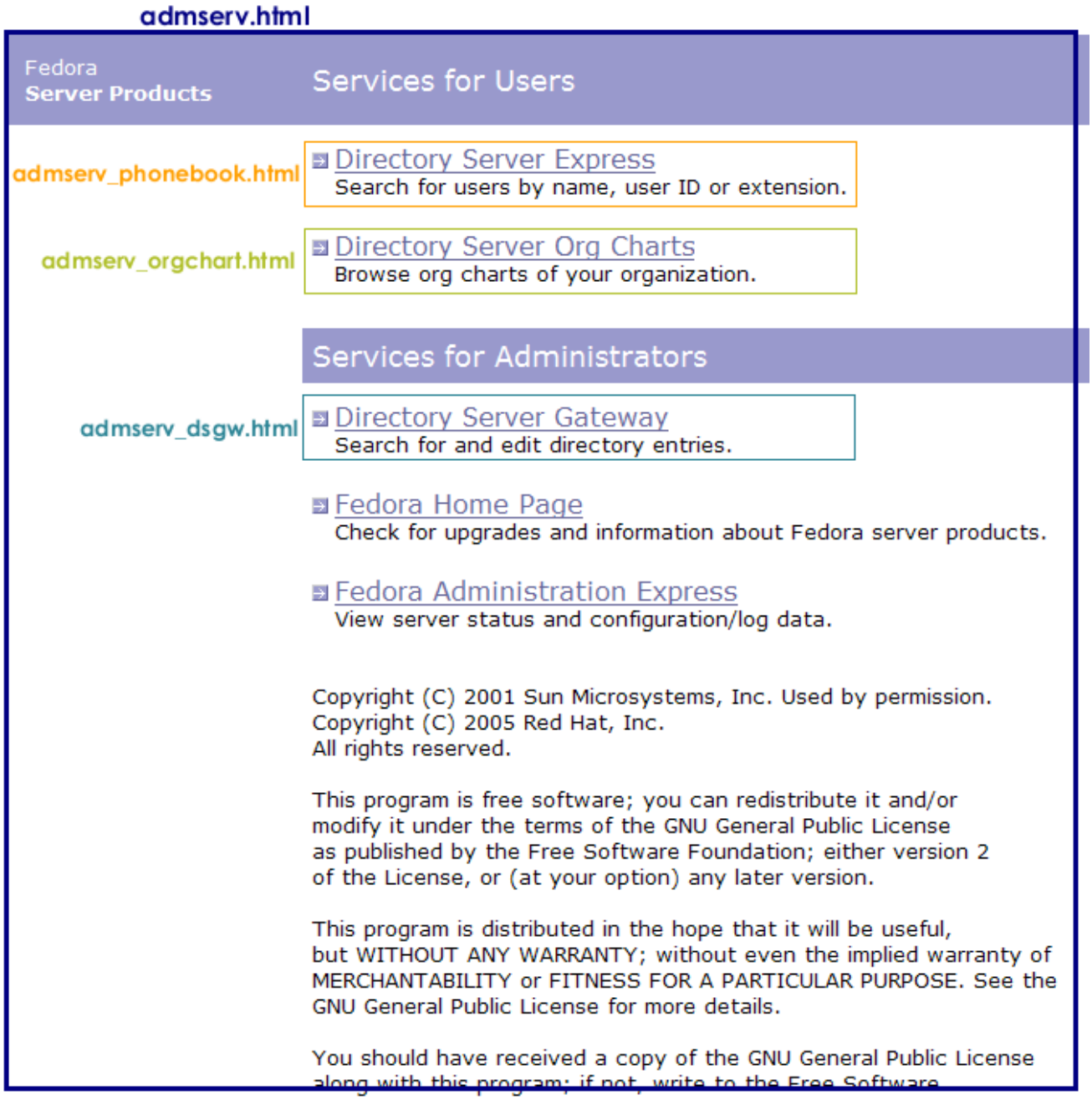


Figure F.4. Intro Page Elements

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**

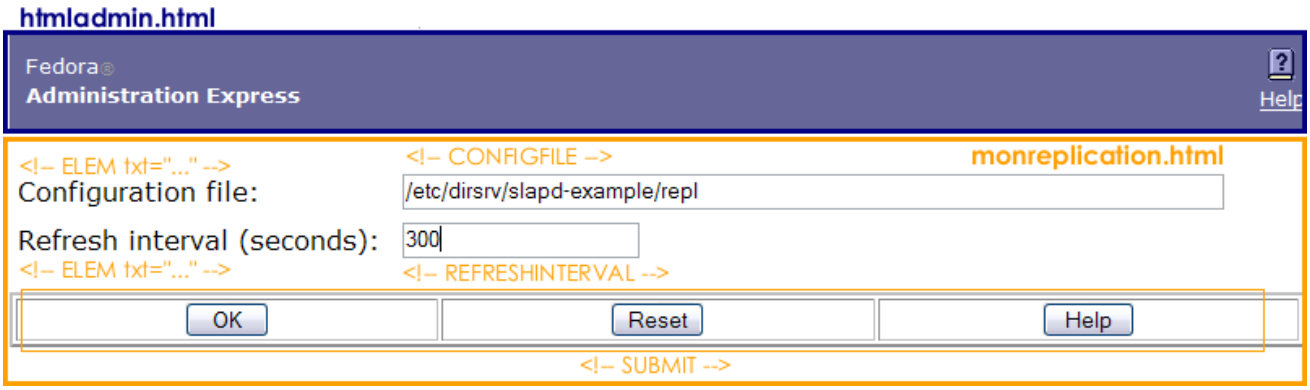


Figure F.5. Monitoring Replication Setup Page Elements

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**

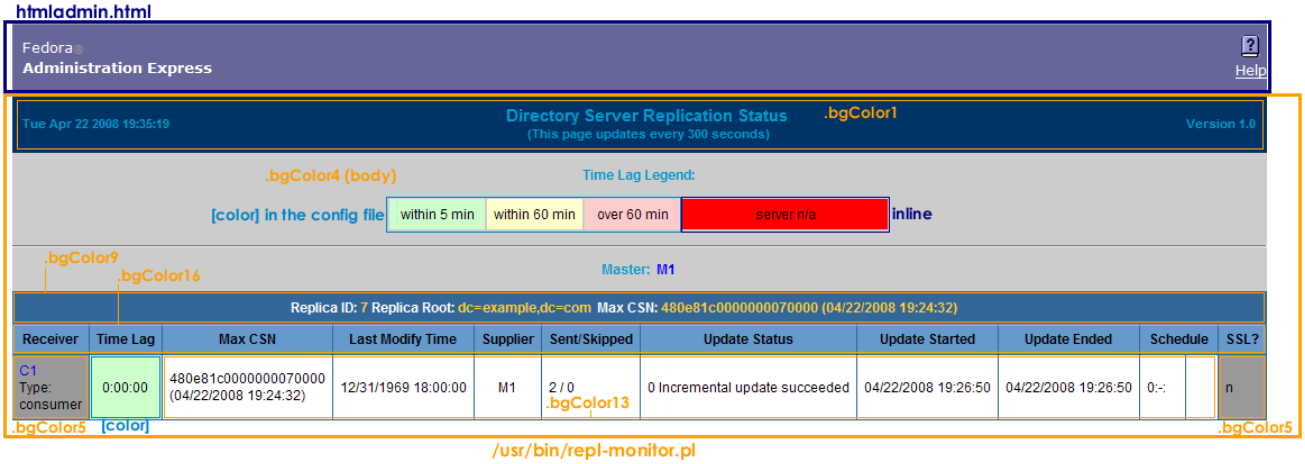


Figure F.6. Monitoring Replication View Page Elements

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**

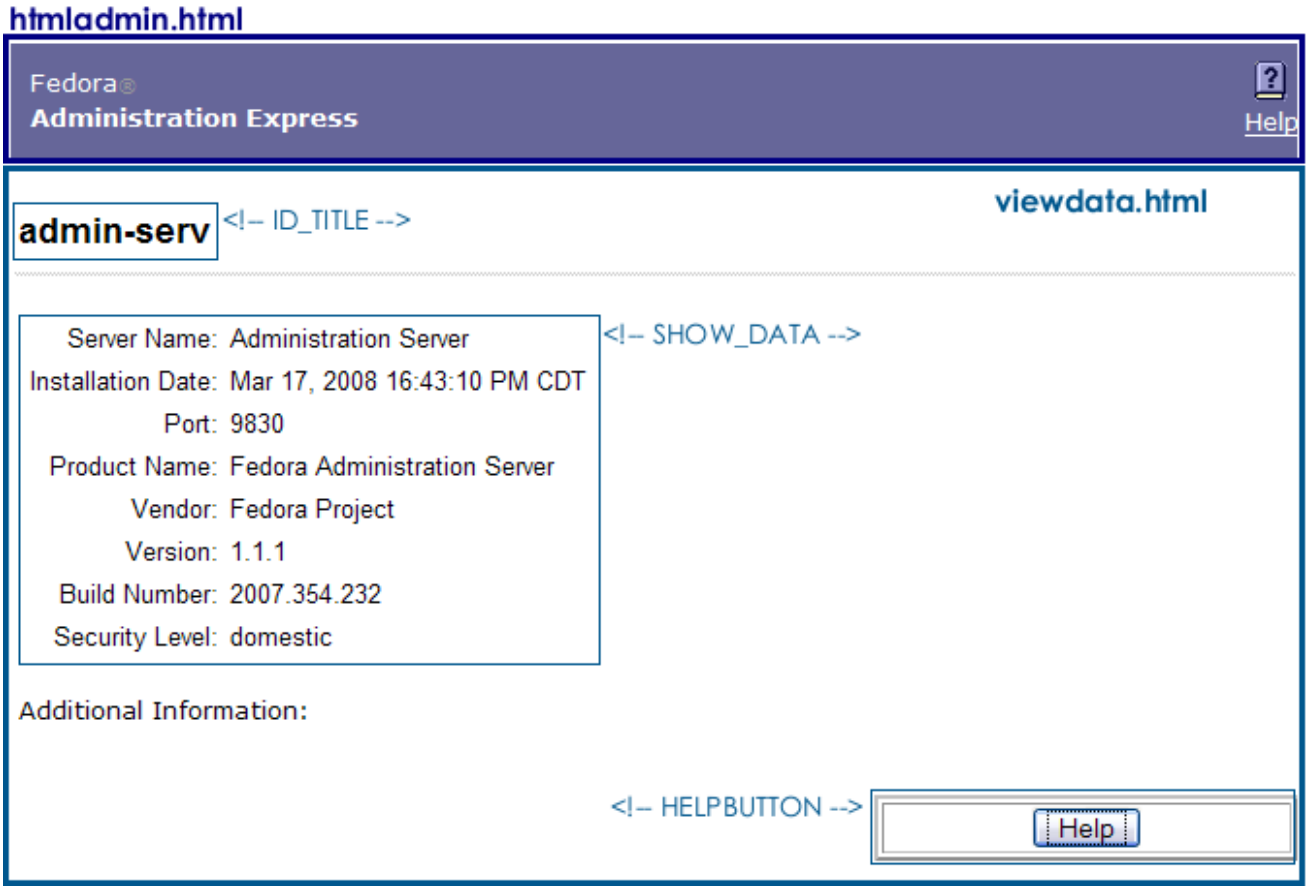


Figure F.7. Server Information Page Elements

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 Administration 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*/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**

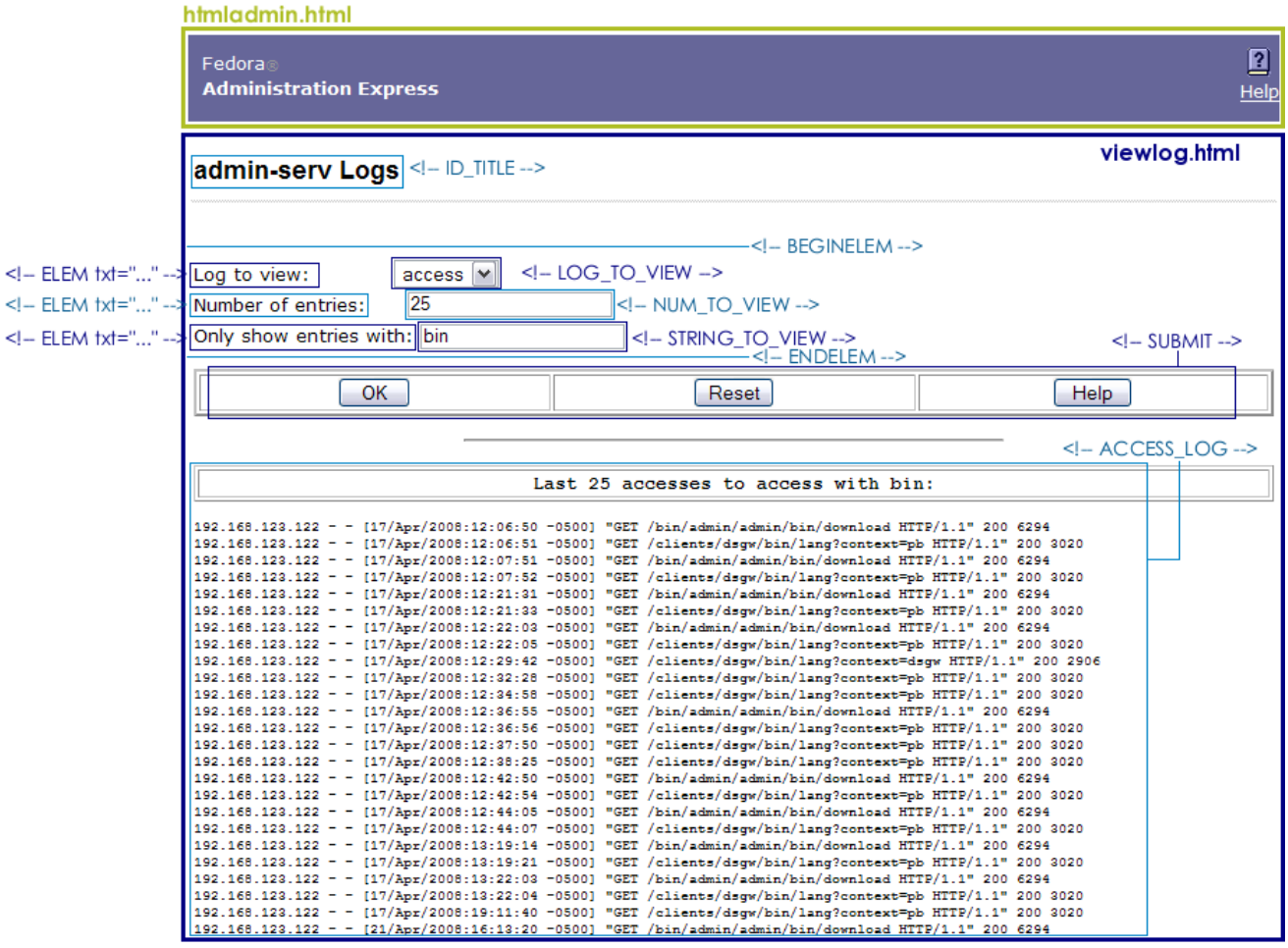


Figure F.8. Log View Page Elements

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
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 -->

Directive	Description	Example
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 -->
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 Administration 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 Administration Server and walks through the Console windows and options.

G.1.1. How the Console, Directory Server, and Administration Server Work Together

The Red Hat Console is an independent Java application which works in conjunction with instances of Red Hat Directory Server and Administration Server. Most server management functions are carried out in server-specific console windows for the Directory Server and Administration Server. Red Hat Console is part of a system that manages Red Hat Directory Server instances and the Administration Server and, therefore, information in the directory. Although Red Hat Directory Server, Red Hat Management Console, and Red Hat Administration 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.

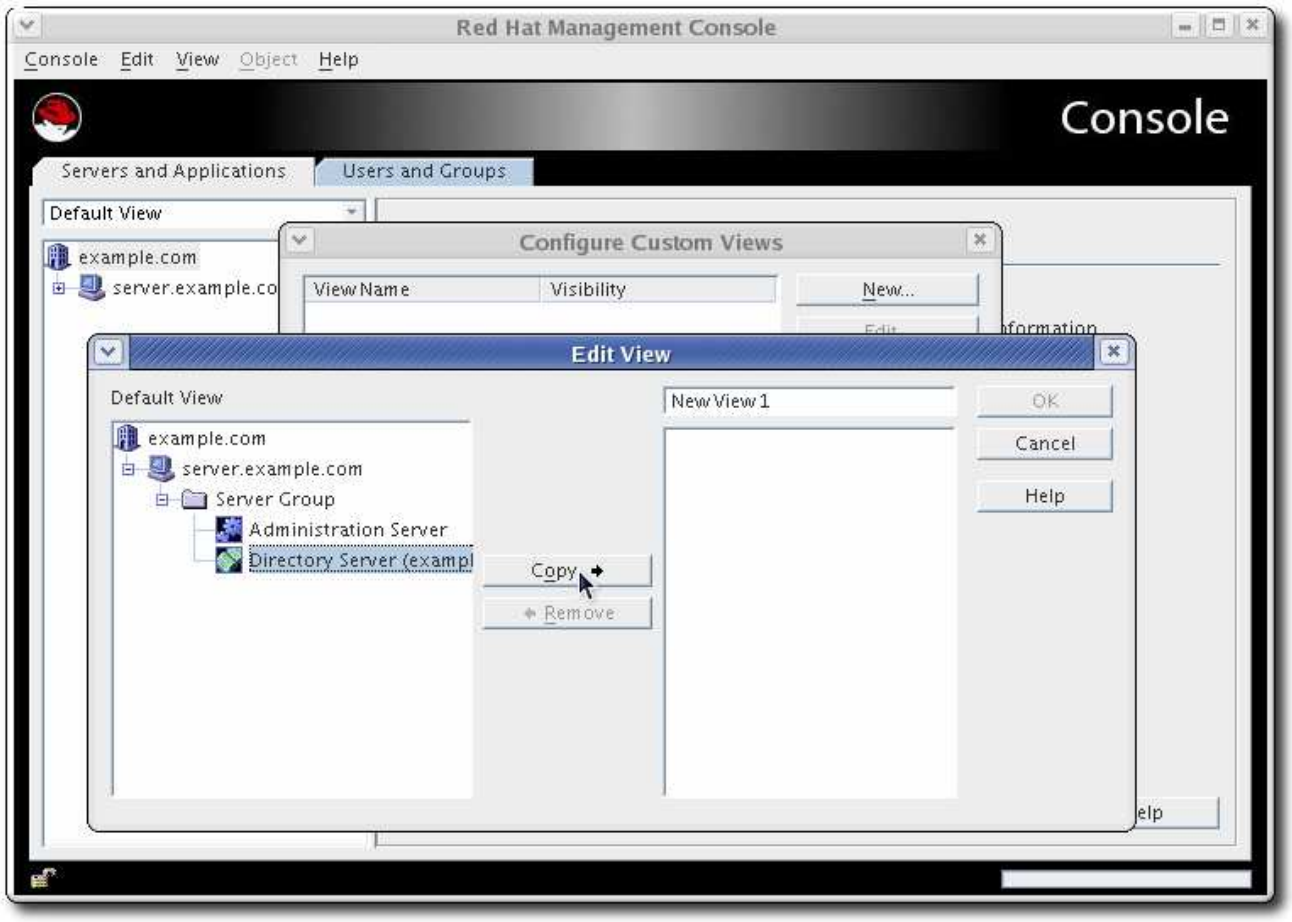


Figure G.1. The Red Hat Management Console Interface

When a user logs into Red Hat Management Console, the Console connects to the Administration Server over Hypertext Transfer Protocol (HTTP). The Administration 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.

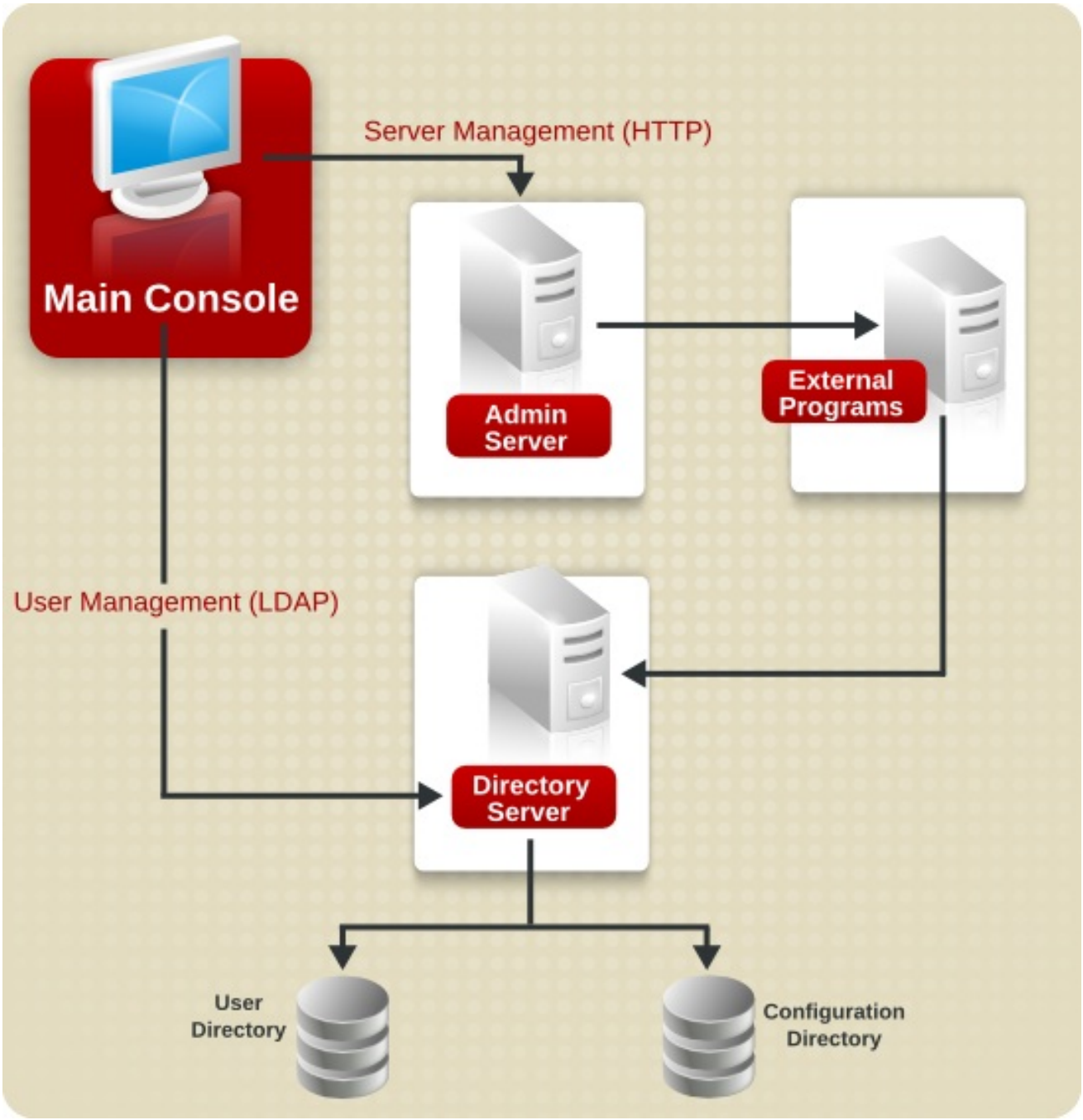


Figure G.2. Simple System Using Red Hat Management Console

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.

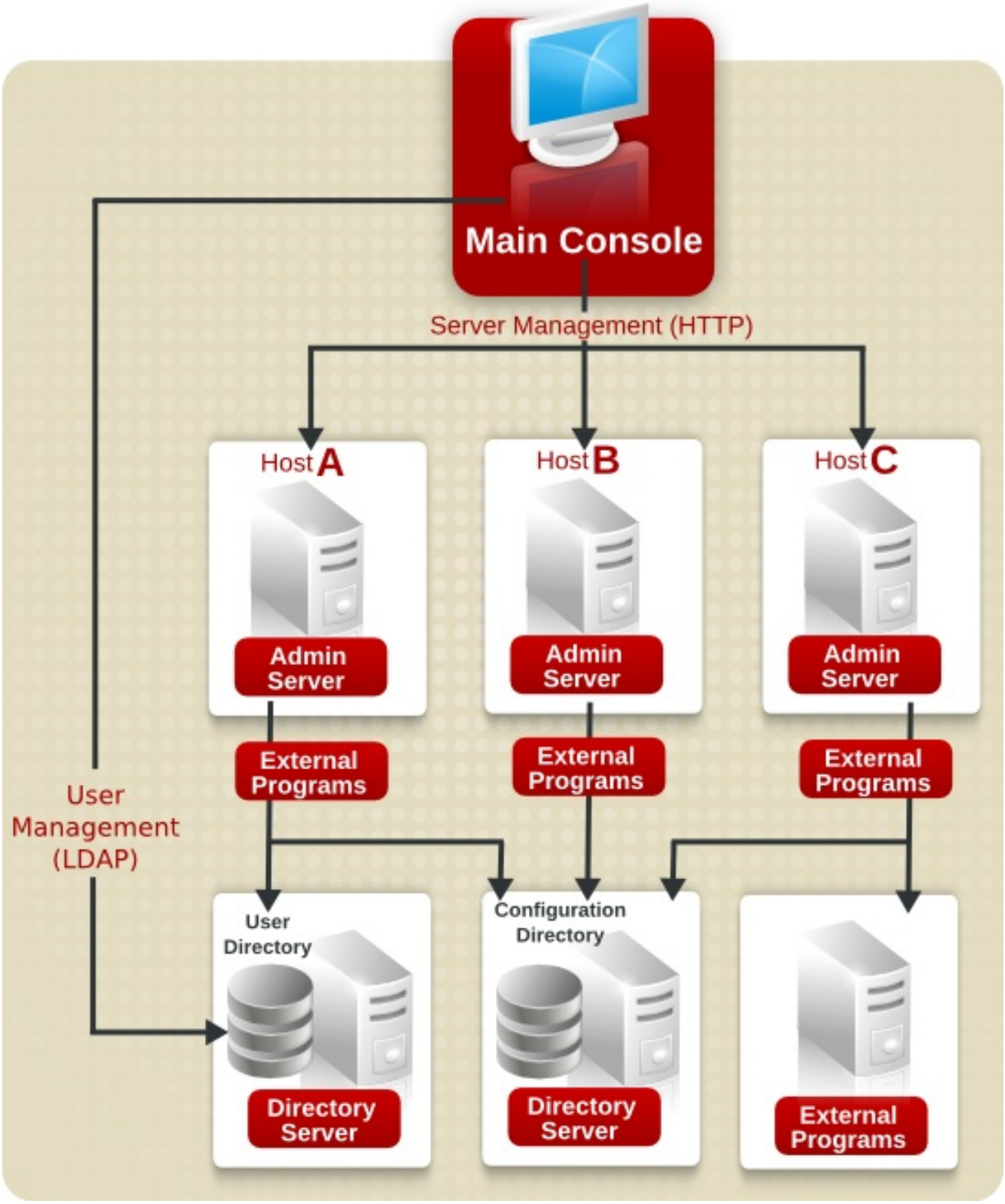


Figure G.3. A More Complex System

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 Administration Server Console) and the types of objects available in that server area.

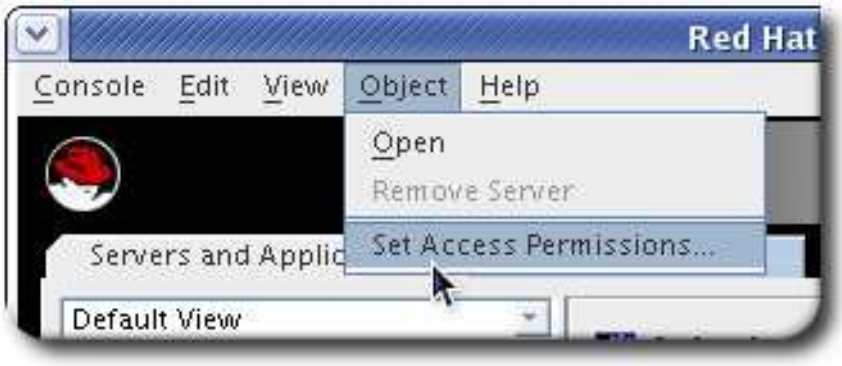


Figure G.4. Main Console Menus

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 Administration 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 Administration 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 Administration 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 Administration Servers and a center information panel. To access the Directory Server instance, directory information, or Administration 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.

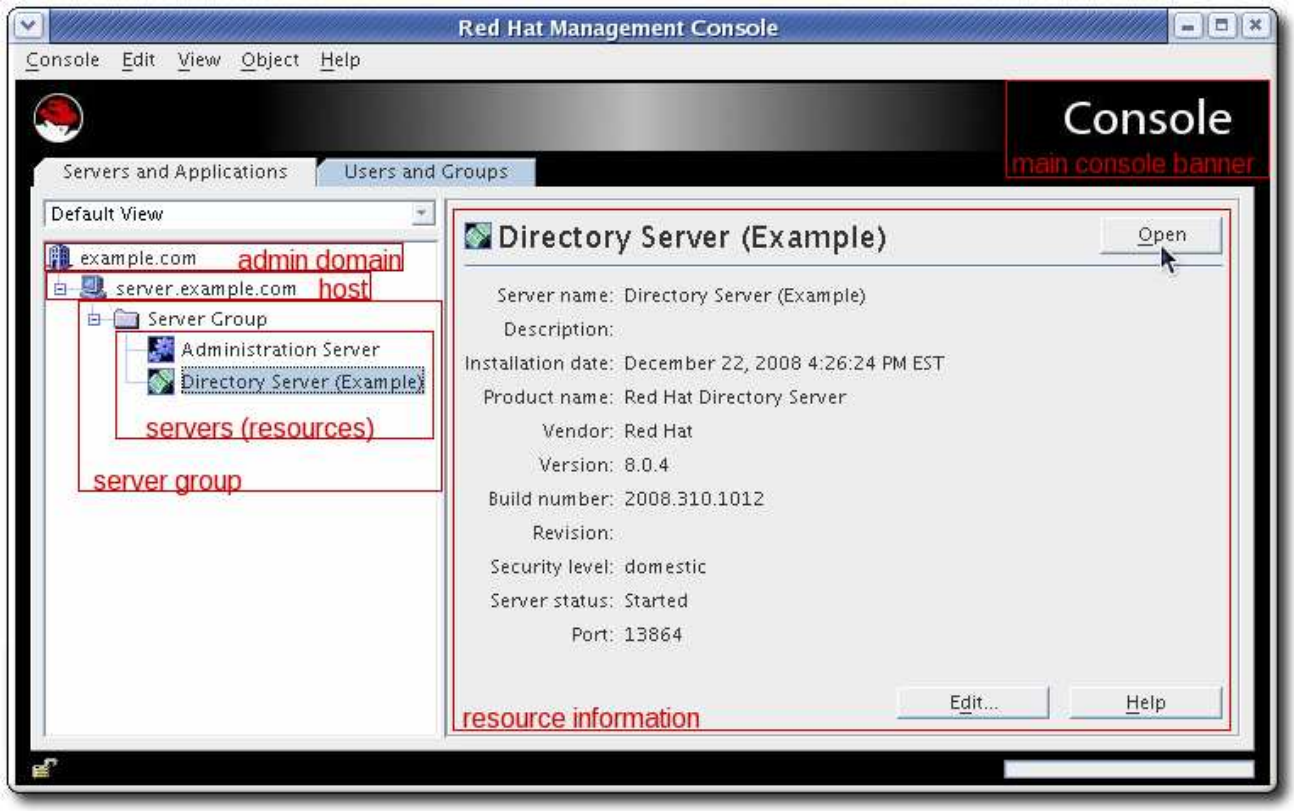


Figure G.5. The Servers and Applications Tab

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 Administration 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 Administration 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.

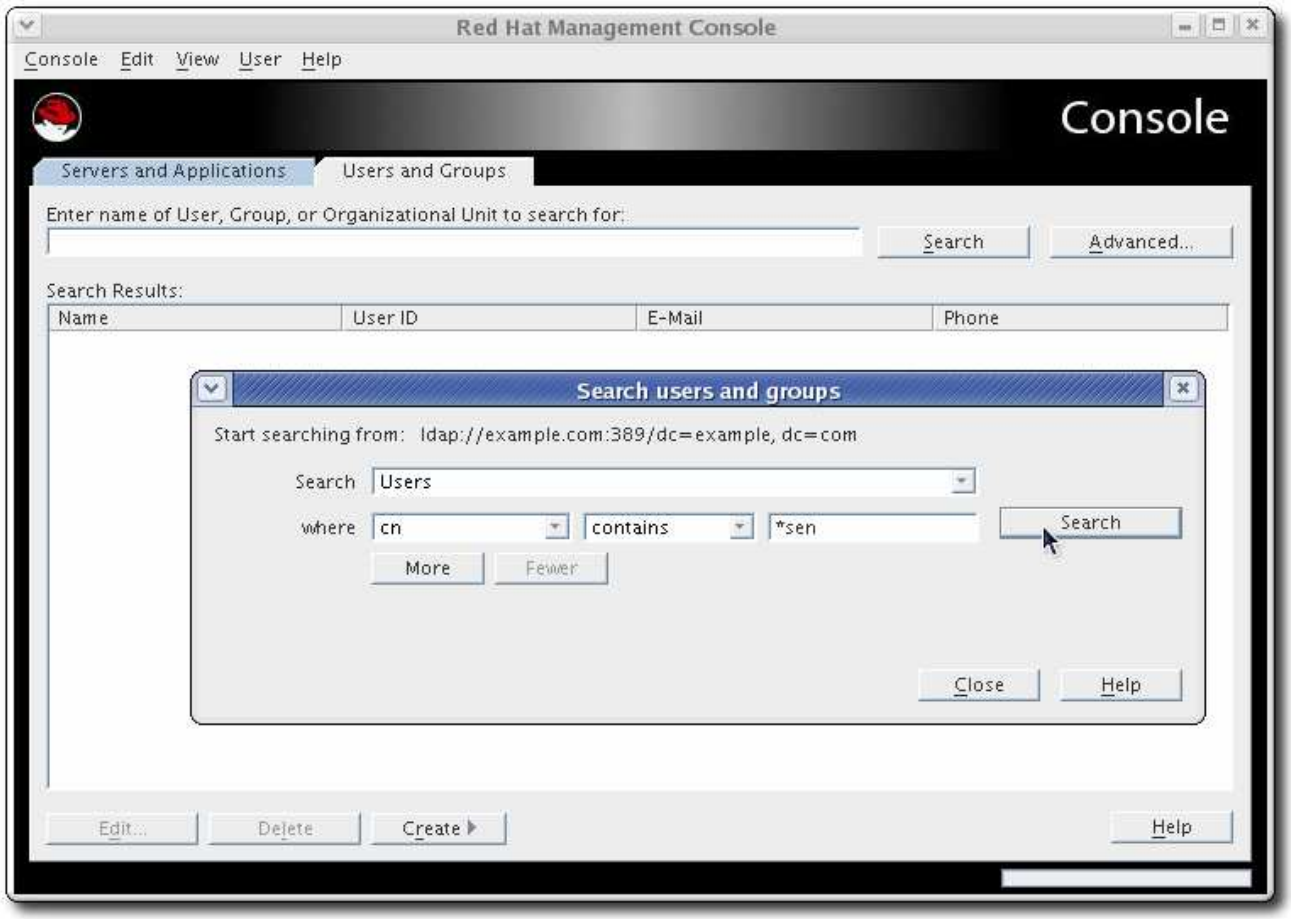


Figure G.6. The Users and Groups Tab

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 Administration 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, TLS 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.

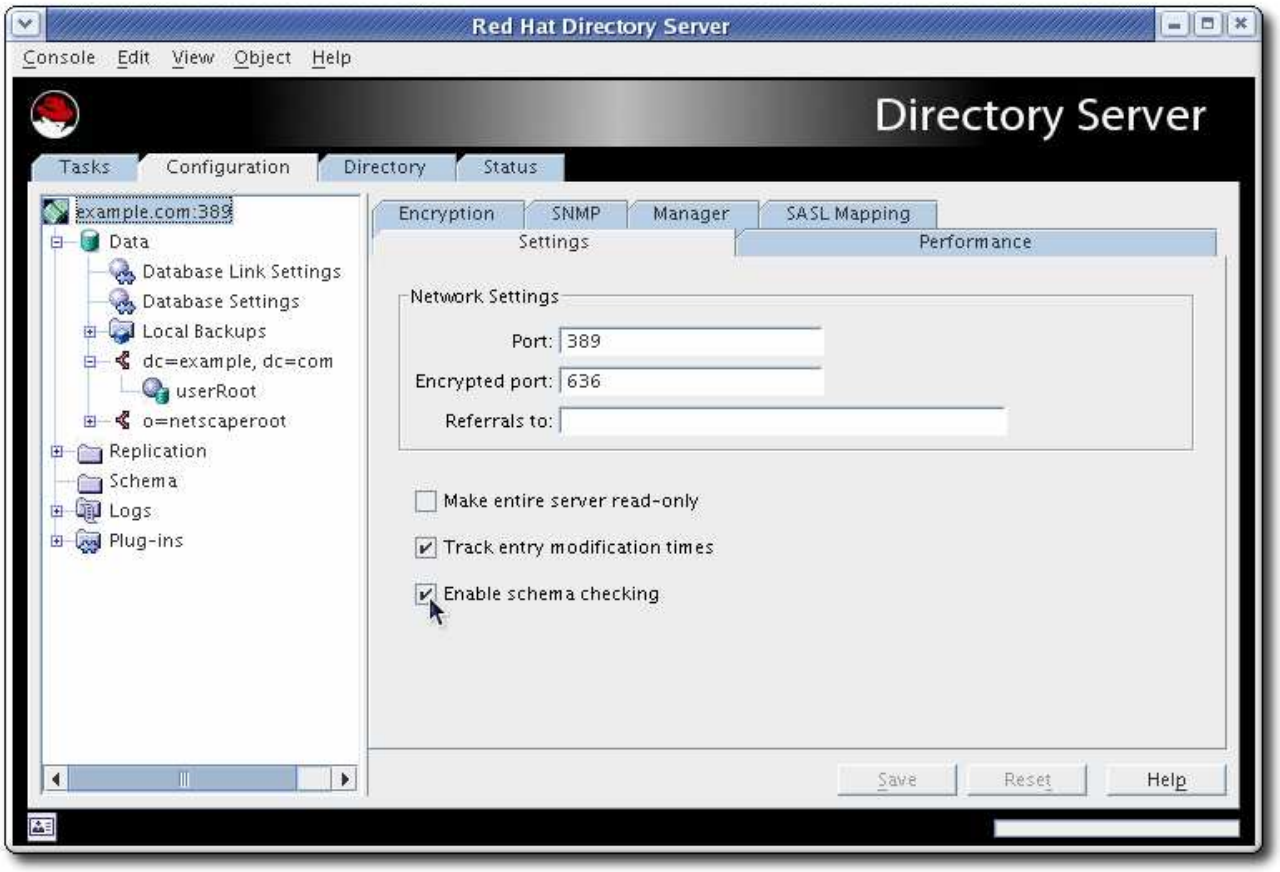


Figure G.7. The Directory Server Console

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 TLS certificates
- **Configuration**, which defines all of the server configuration settings, including SASL and TLS 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 *Red Hat Directory Server Administration Guide*.

G.1.4.2. The Administration Server Console

The Administration Server itself administers the configuration of other servers, especially the configuration and user directories for the server group. The Administration Server Console manages the Administration 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 Administration Server configuration for the server to properly manage those servers.

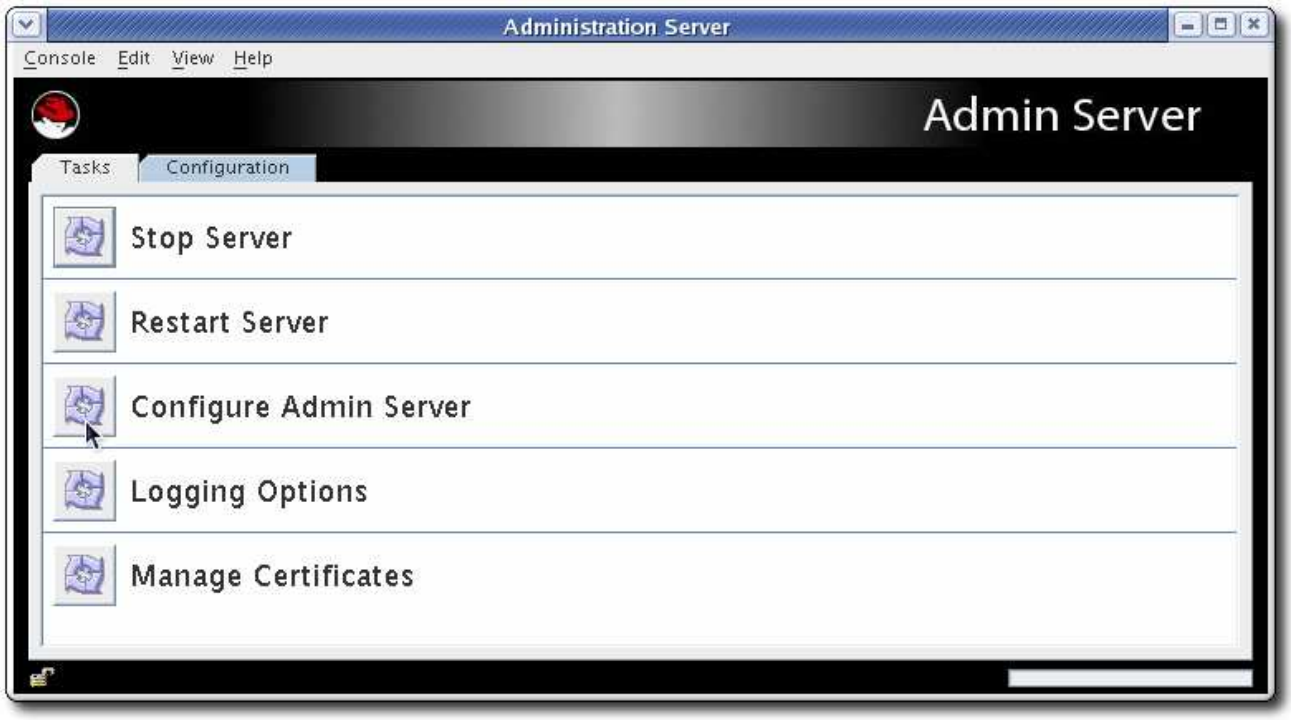


Figure G.8. The Administration Server Console

The Administration 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 Administration Server instance, setting up logging, and managing TLS certificates
- **Configuration**, which defines all of the Administration Server configuration settings, including TLS authentication, port numbers, and logging, as well as the Configuration Directory Server and User Directory Server settings which the Administration Server uses to connect to the directory services

The procedures for using the Administration Server Console to manage the Administration Server configuration and associated directory services is covered in the *Using the Admin Server* guide.

G.2. 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.2.1, "Changing Profile Locations"
- Section G.2.2, "Restoring Default Font Settings"
- Section G.2.3, "Changing Console Fonts"
- Section G.2.4, "Reordering Table Columns"

- Section G.2.5, "Customizing the Main Window"

G.2.1. Changing Profile Locations

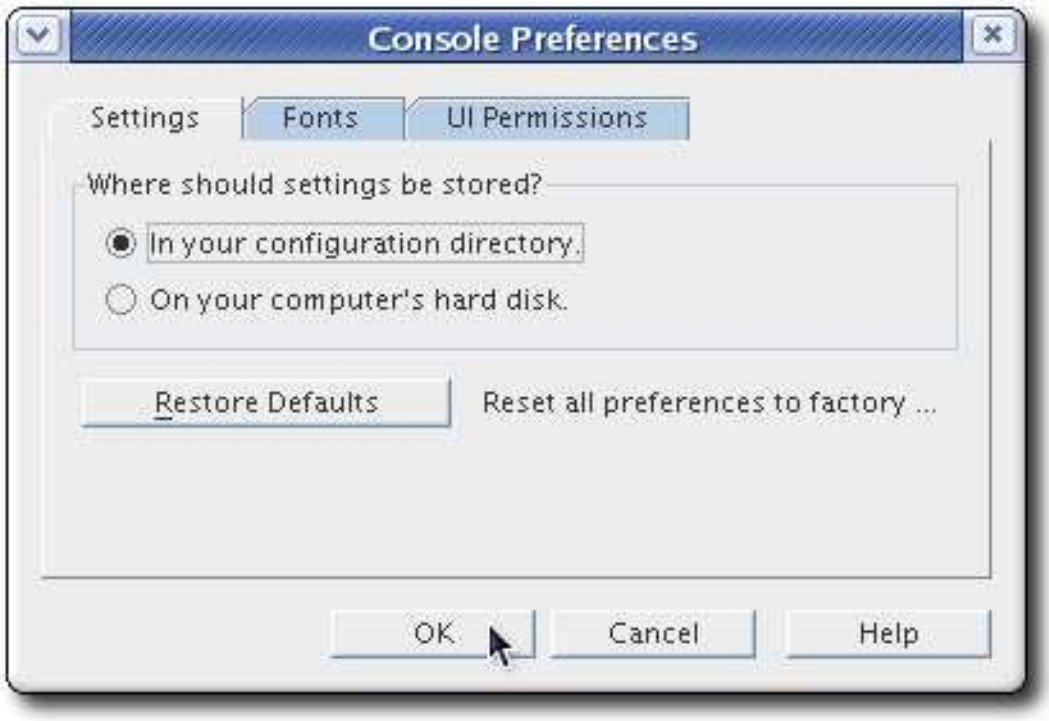
The Console formatting is stored in *profiles*. An entry's profiles can be stored locally, which means that they are only available at a specific workstation, or can be stored in the configuration directory, so they are accessible anywhere.

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.

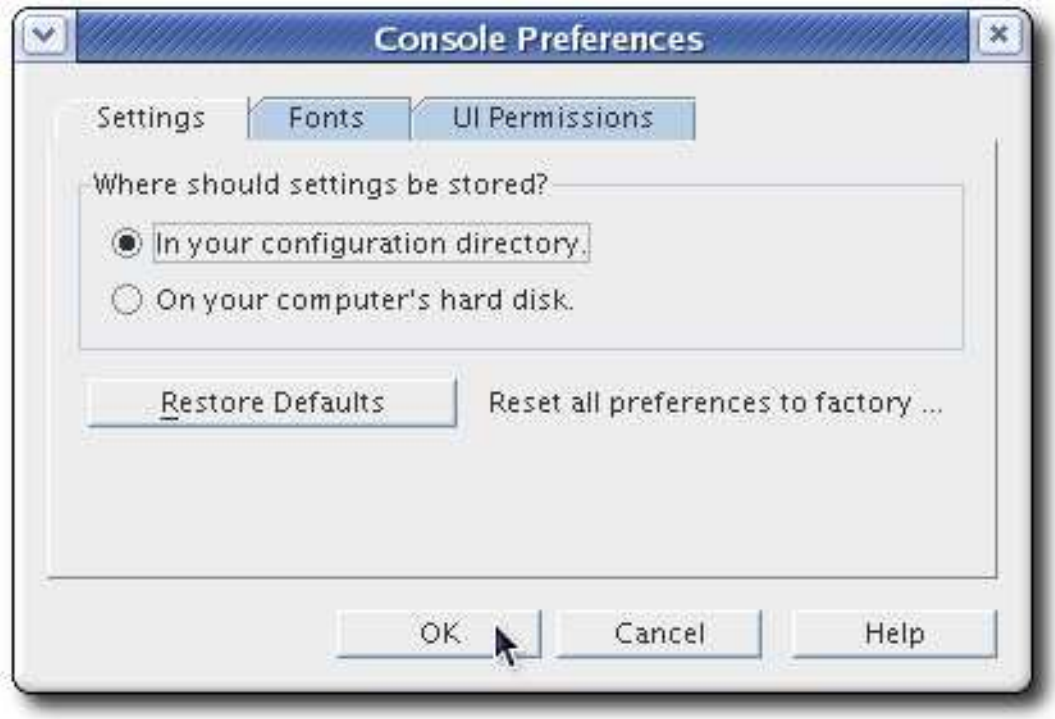
4. Click **OK**.

G.2.2. 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.2.3. 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**.

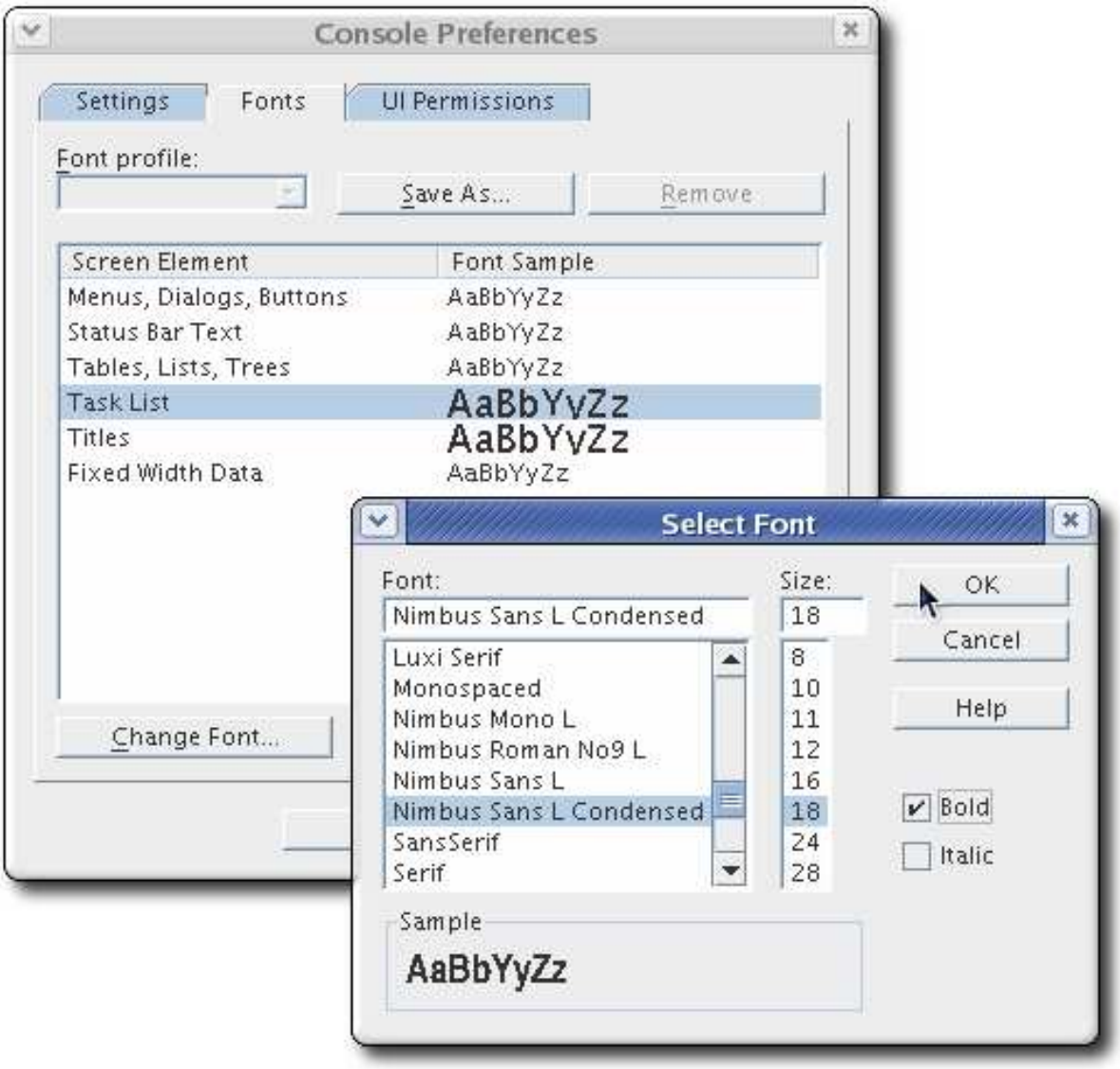
2. Click the **Fonts** tab.

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.

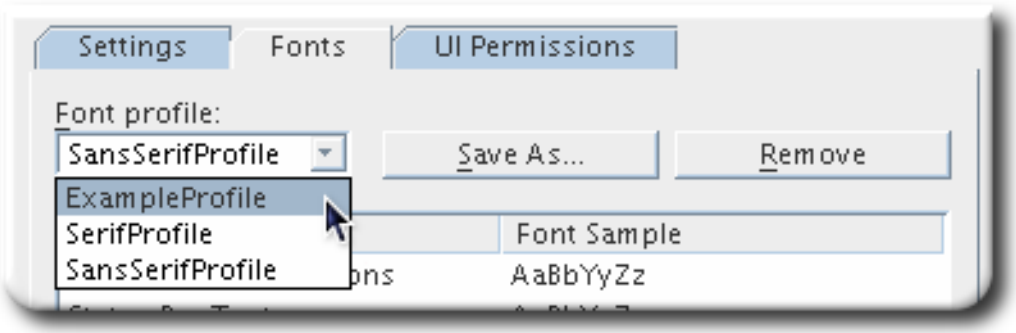
5. Edit the font for that specific element. There are three settings which can be changed: the font family, the size, and the formatting (bold or italic).



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**.

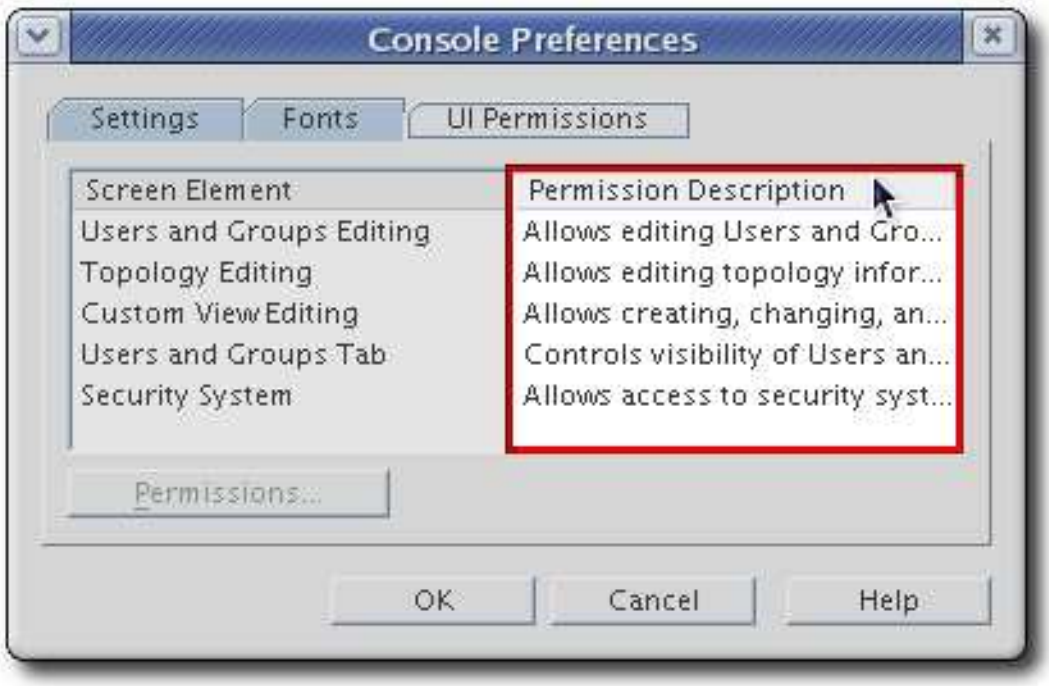


To delete a font profile, simply make sure that it is selected from the drop-down menu in the **Fonts** tab, and click the **Remove** button.

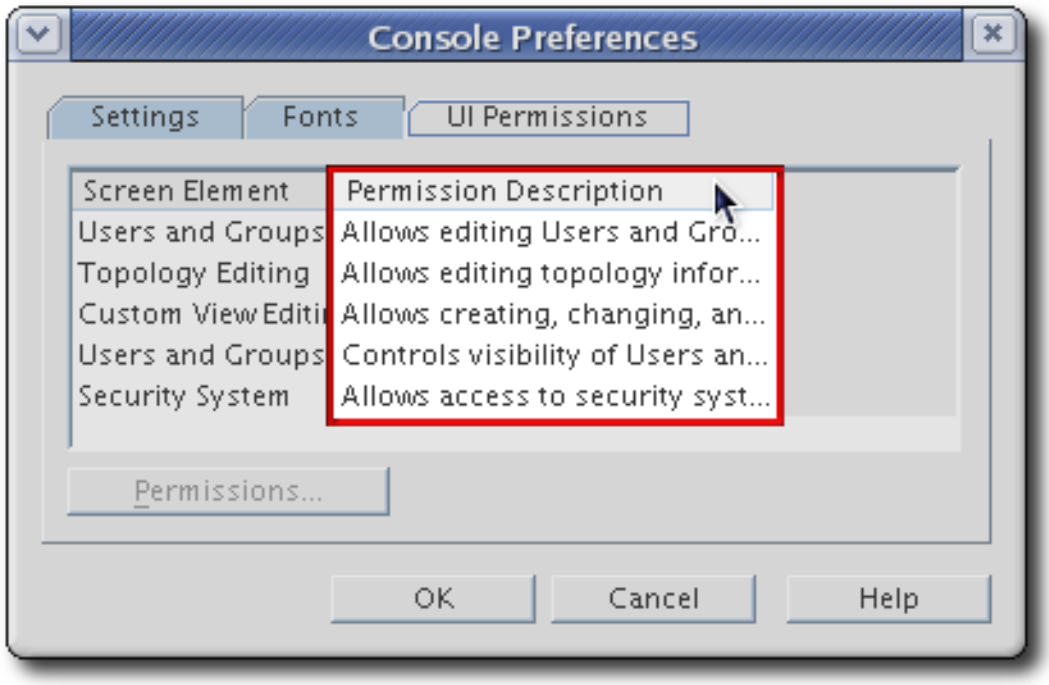
G.2.4. Reordering Table Columns

The columns in a table can be rearranged by dragging them into a new position.

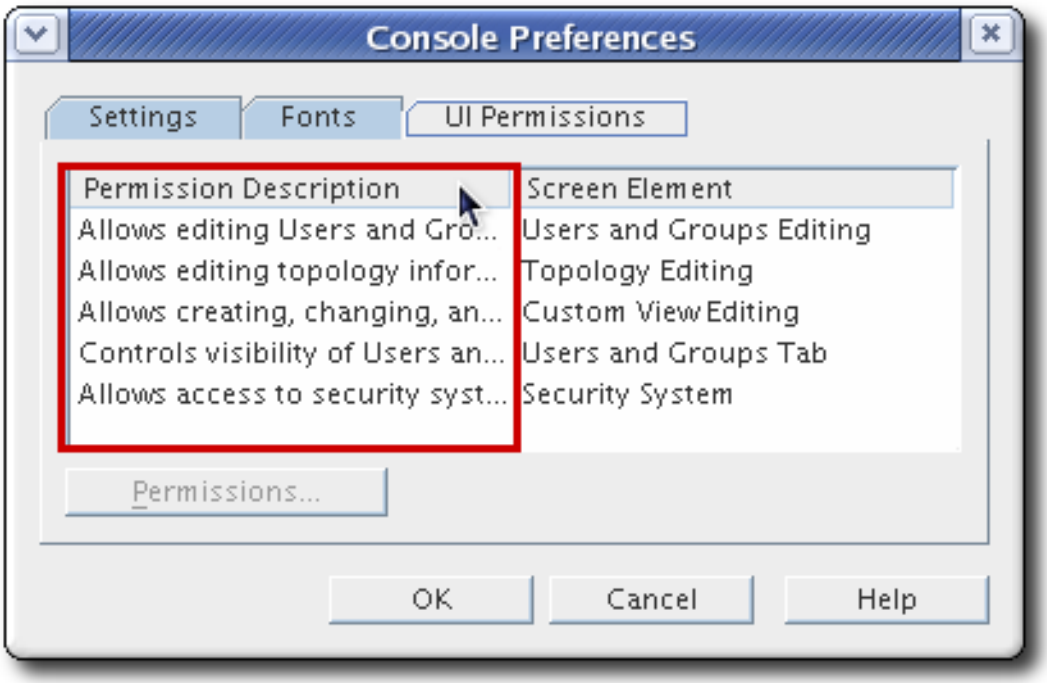
1. Click in the table heading.



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.

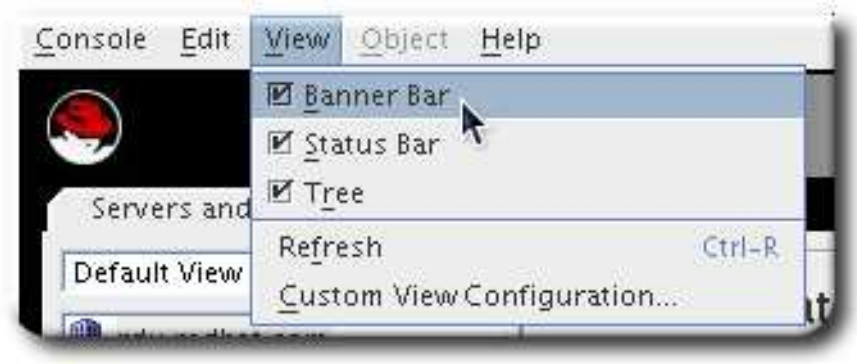


3. When you release the mouse button, the column snaps into its new position.

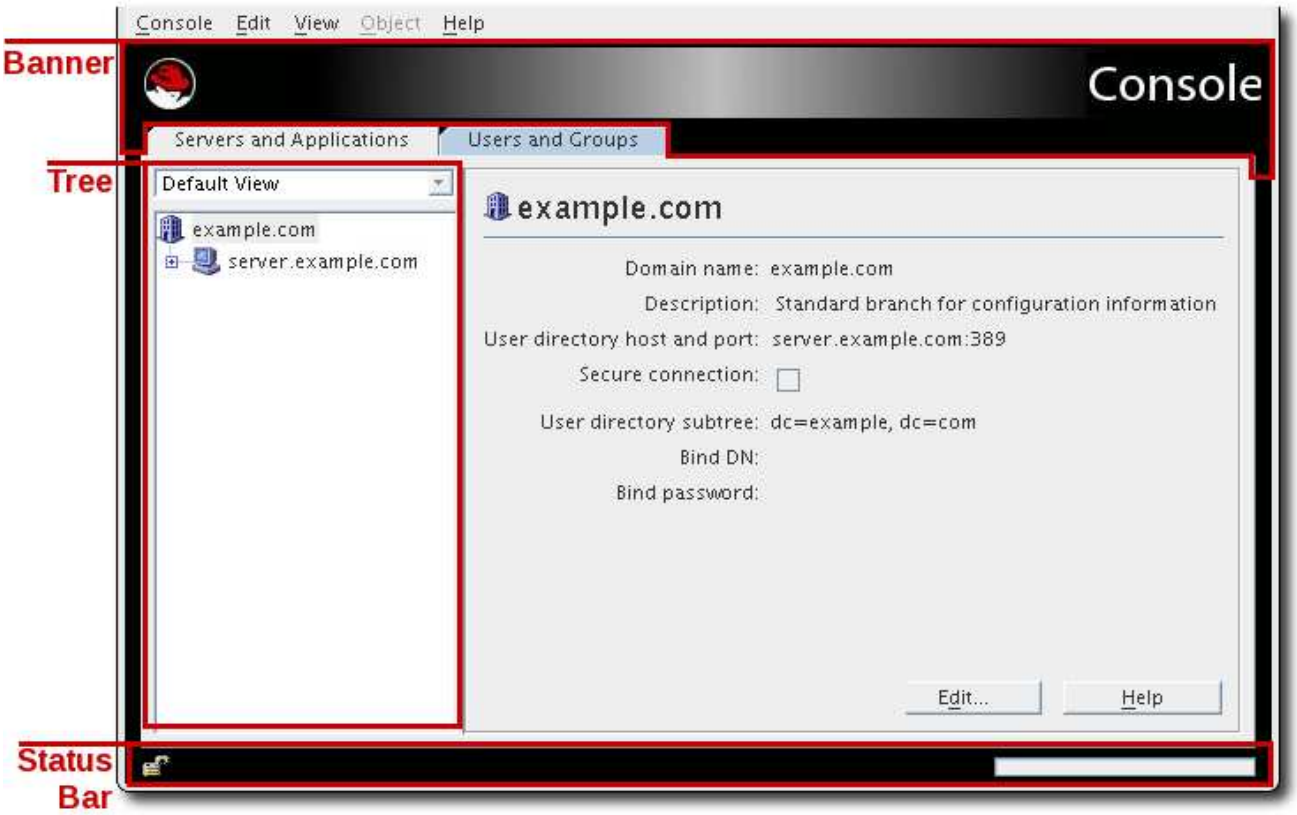


G.2.5. Customizing the Main Window

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.



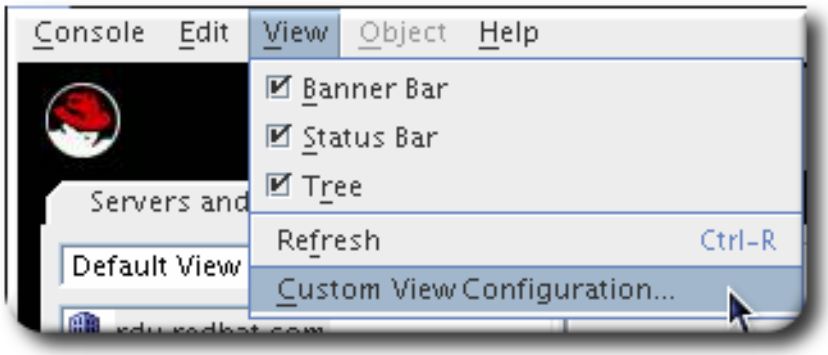
G.2.6. Working with Custom Views

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.

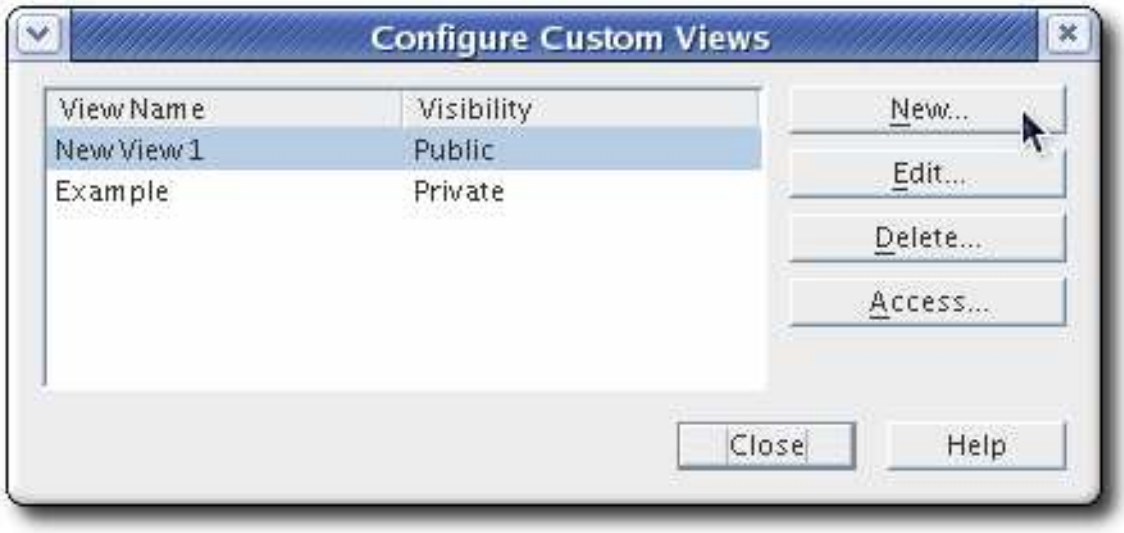
G.2.6.1. Creating Custom Views

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**.



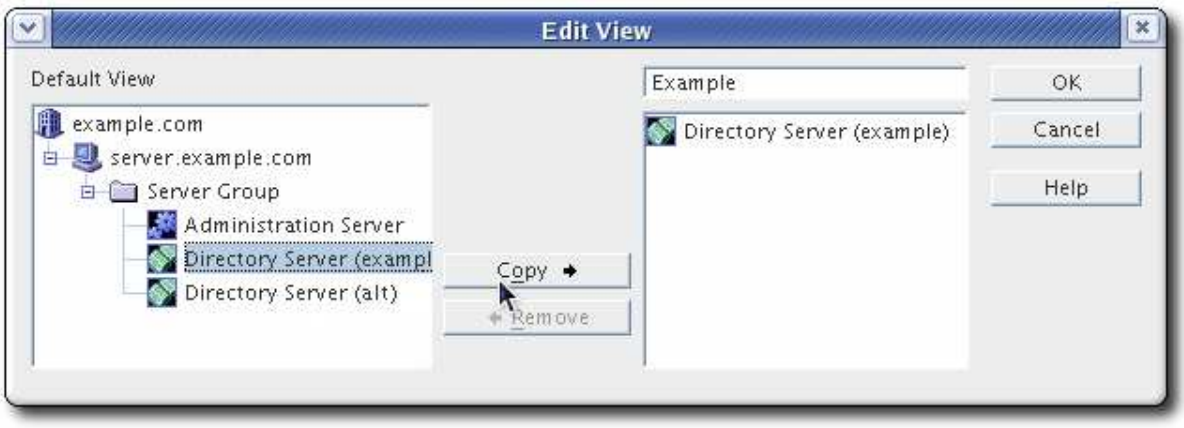
3. Choose whether the new view will be public or private, then click **OK**.

   - 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.2.6.3, "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.



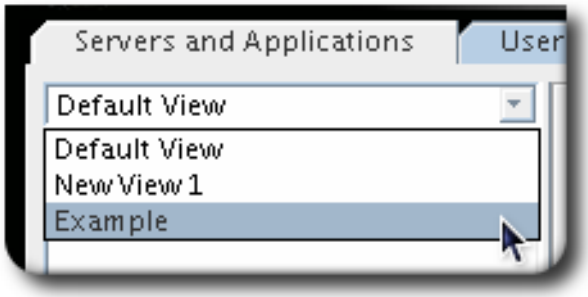
To select a range of resources, click the **SHIFT** key and select the first and last entries; select multiple, separate resources by holding down the **Ctrl** key and selecting the entries.

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.2.6.2. Switching to a Custom View

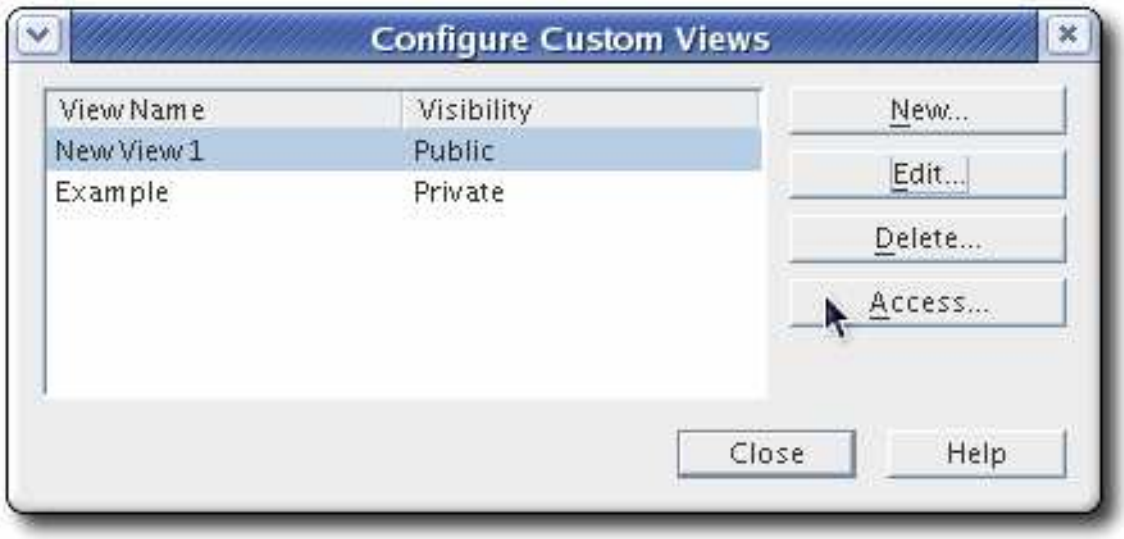
Choose the required custom view from the drop-down list on the **Servers and Applications** tab.



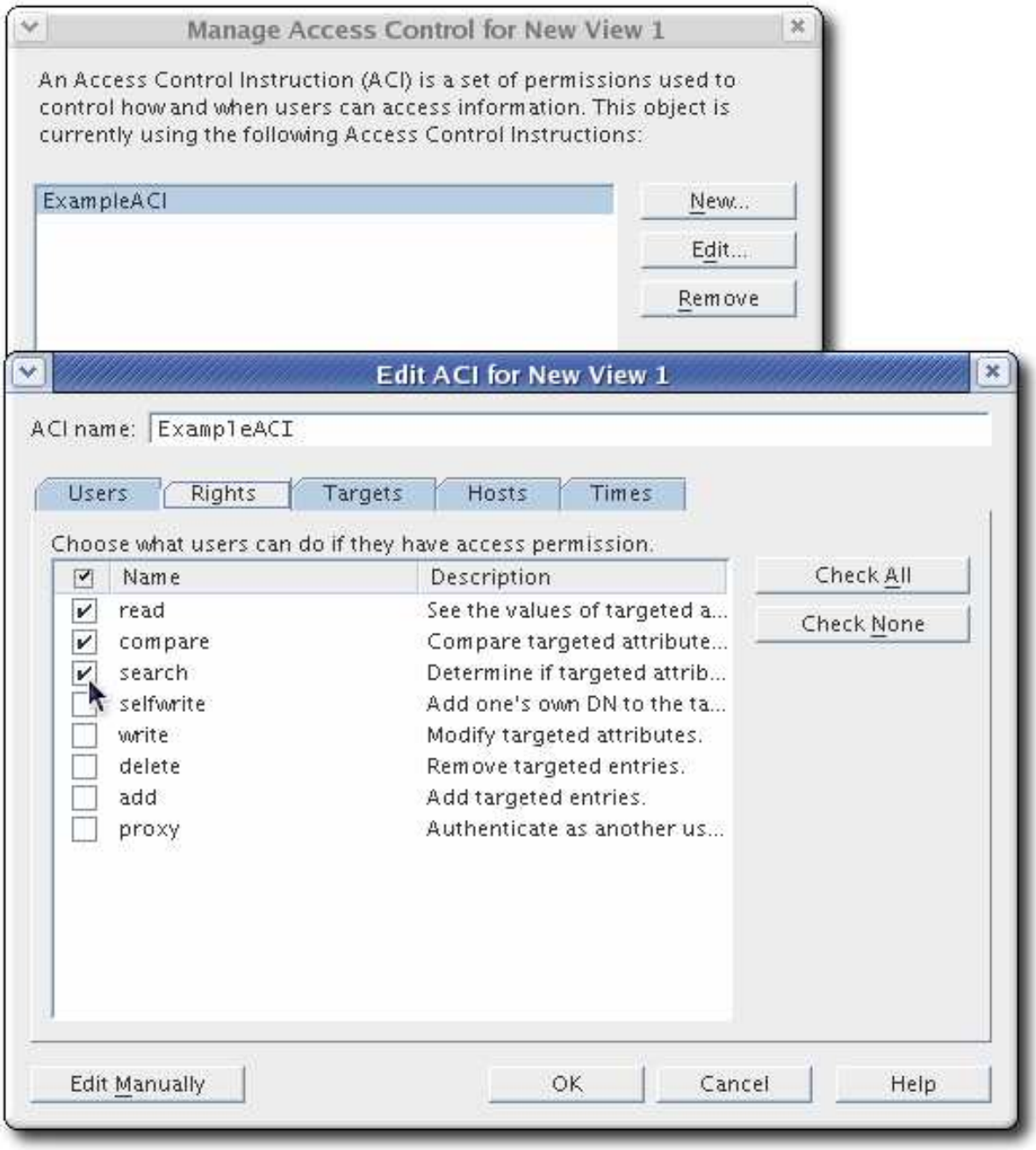
To return to the default view, choose **Default View** from the drop-down list.

G.2.6.3. 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

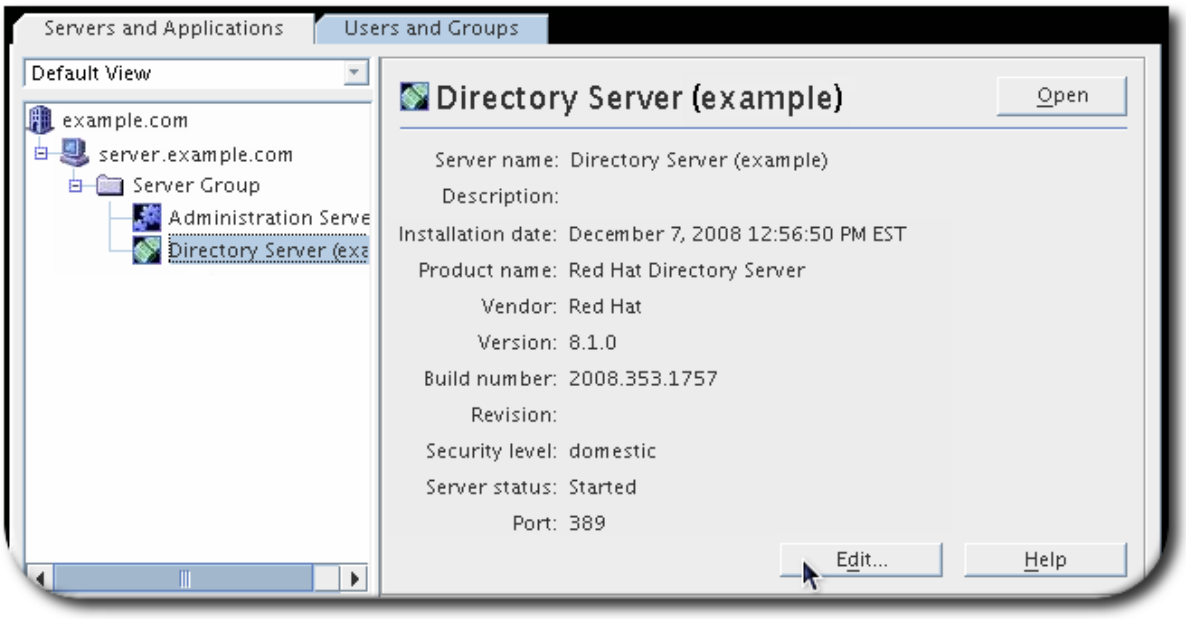
The server instances managed by the Red Hat Management Console are arranged in a hierarchy. At the top is the admin domain. Within the domain are hosts, representing different server machines. Each host has server groups, which identifies an inter-related group of Directory Servers using the same Administration Server instance. The individual Directory Server instances and a single Administration Server instance belong within a server group. There can only be one Administration 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

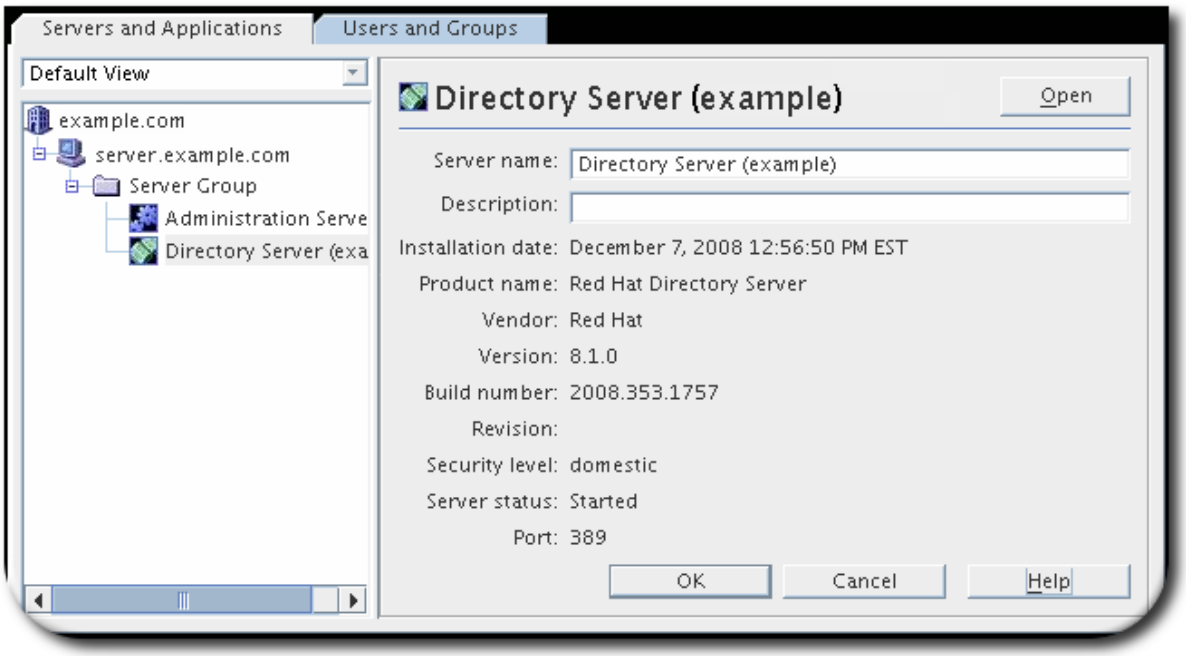
The Red Hat Console displays some information about every admin domain, host, group, and server instances. Most of this information — such as the installation date and build number — are not editable, but some information is.

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**.

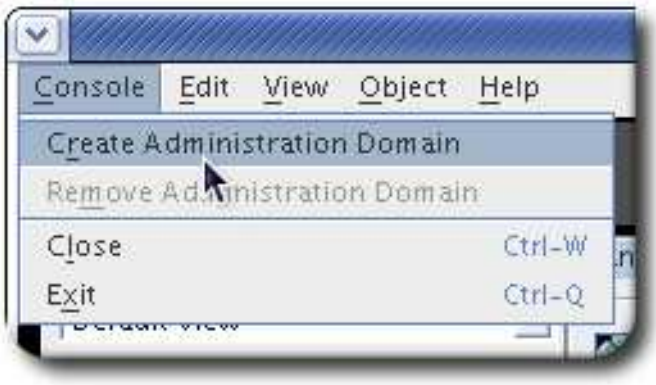
G.3.2. Creating and Removing Admin Domains

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 Administration 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.4. MANAGING DIRECTORY SERVER USERS AND GROUPS

Users for both multiple Red Hat Directory Server instances and Administration 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.

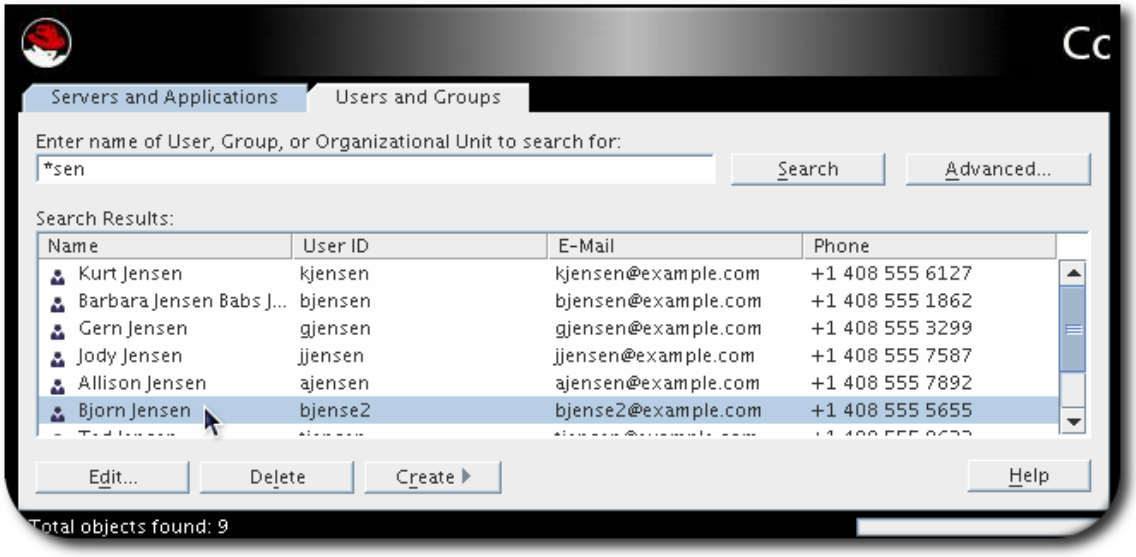
Section G.5, "Setting Access Controls" describes how to work with user and group information when setting access privileges and other security information.

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 Administration 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 (*).

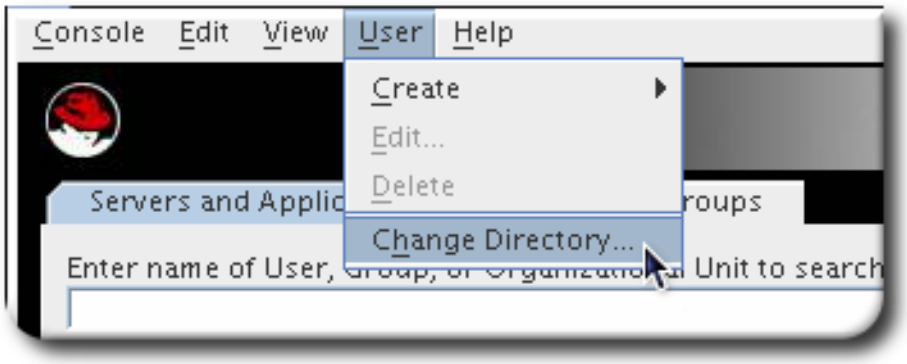


    - For a more complex or focused search, click the **Advanced** button, and enter the attributes to search (such as ***cn***, ***givenname***, or ***ou***), the kind of search, and the search term. To add or remove search criteria, click the **More** and **Fewer** buttons.

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.

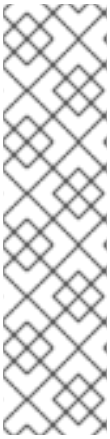
   - **User Directory Host**. The fully qualified host name for the Directory Server instance.
   - **User Directory Port** and **Secure Connection**. The port number for the connection and whether this is a TLS (LDAPS).
   - **User Directory Subtree**. The DN of the subtree to search in the directory; 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 *Red Hat Directory Server Administration Guide*.

G.4.2.1. Directory and Administrative Users

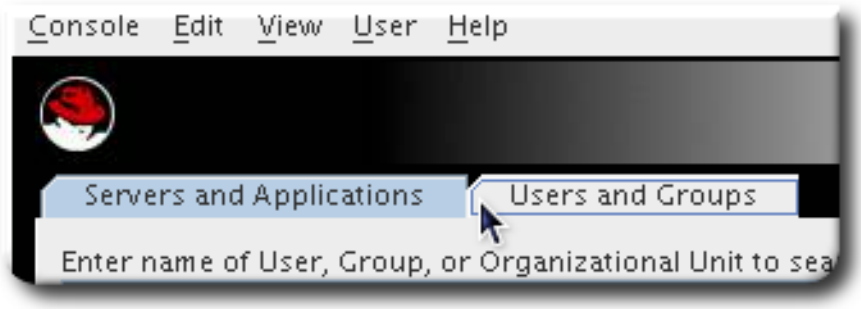


NOTE

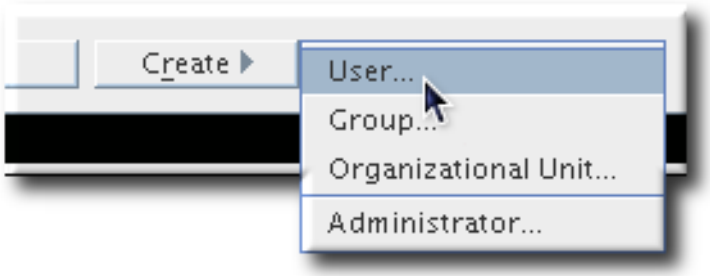
A user can be added to the Directory Server user database through the Console or a user can be added as an Administration Server administrator. The process is almost identical, with two exceptions:

- 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.

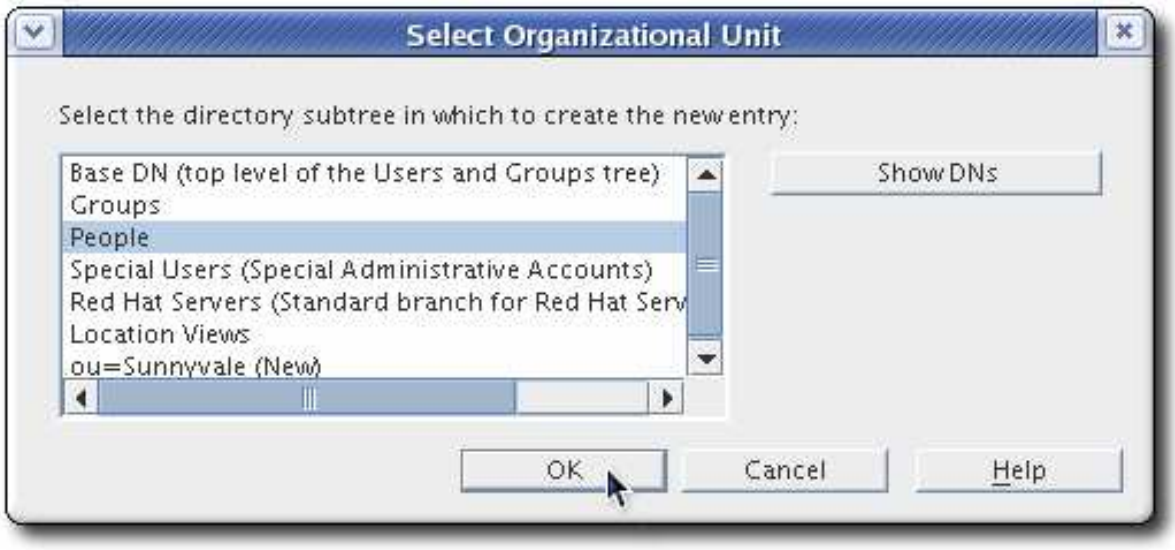


2. Click the **Create** button, and choose **User**.



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

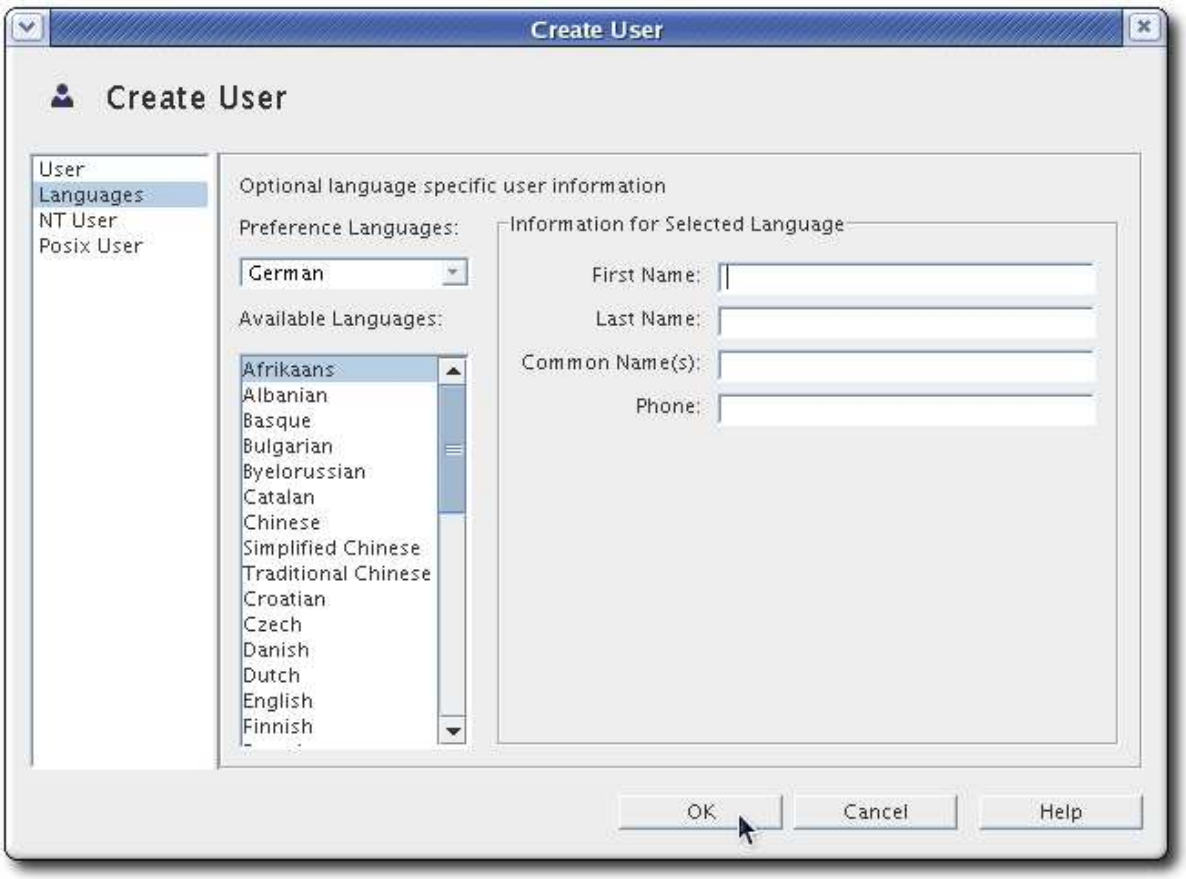
When creating an administrator, there is no option to select the ***ou*** to which to add the user as there is with a regular Directory Server user. This is because the administrator is added to **ou=Groups,ou=Topology,o=NetscapeRoot**, with the admin users.

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:

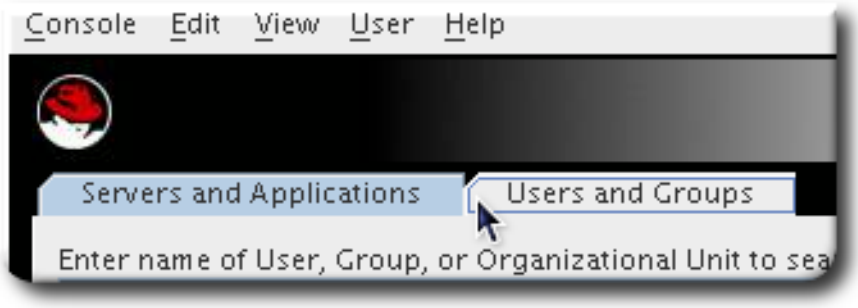
- A *static group* has members who are manually added to it, so it is *static* because the members do not change unless an administrator manually adds or removes users.

- 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.

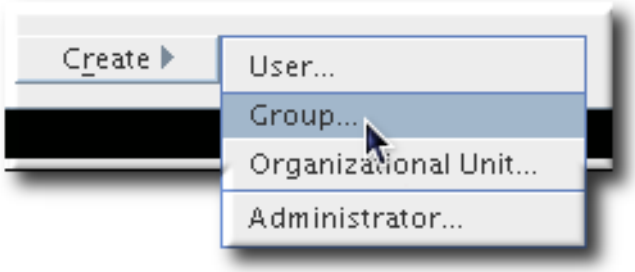
- A *certificate group* includes all users who have a specific attribute-value pair in the subject name of the certificate. For example, the certificate group could be based on having the string **st=California,ou=Sales,ou=West** in the subject name. If a user logs onto a server using a certificate with 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.

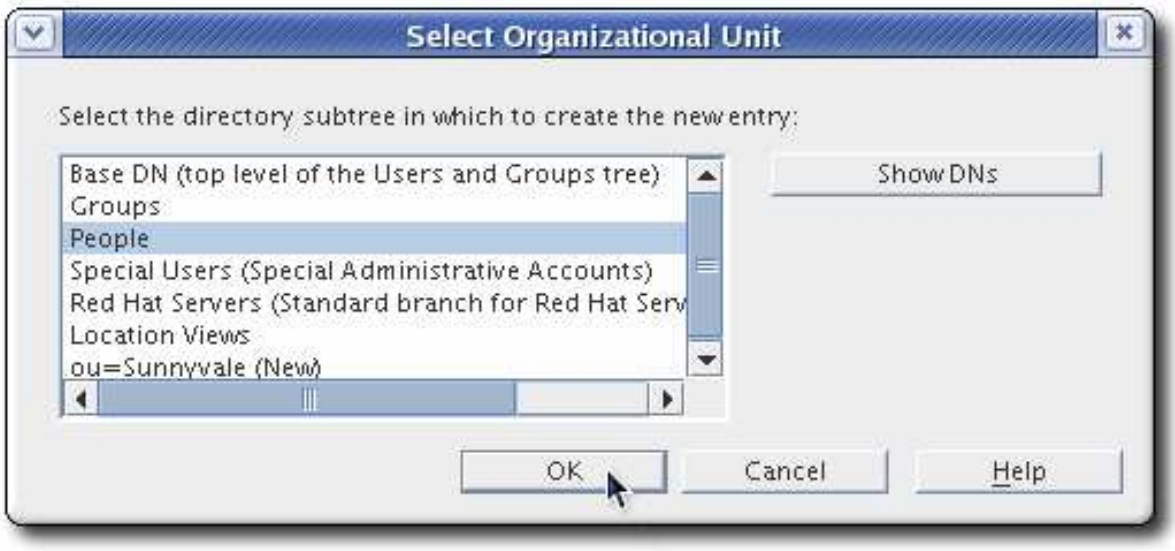


2. Click the **Create** button, and choose **Group**.



   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.



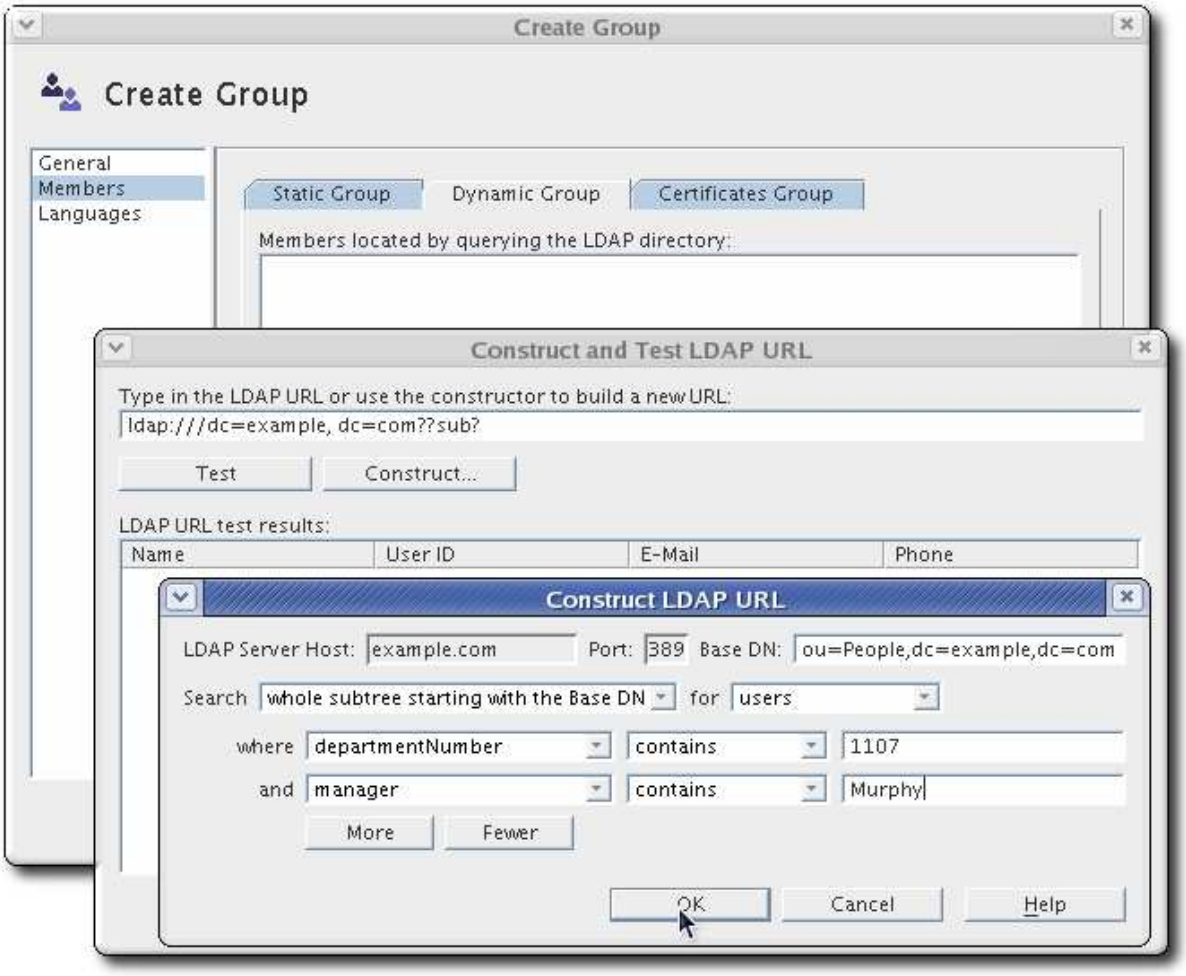
   The subtree entry can be an ***ou*** or a view, if views have been added to the directory.

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.



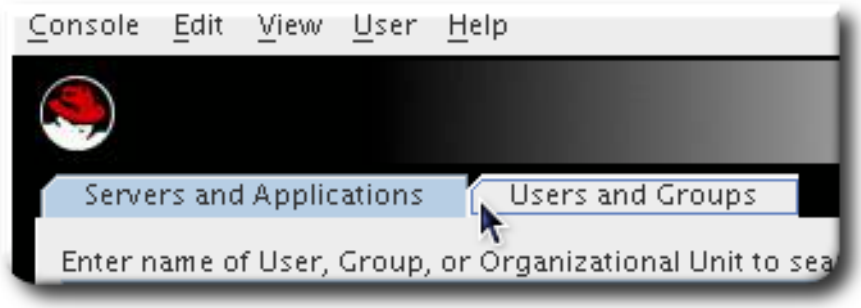
NOTE

The different kinds of groups and how to configure their members are explained in more detail in the *Red Hat Directory Server Administration Guide*.

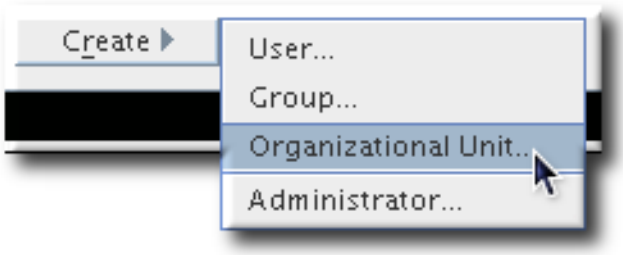
G.4.2.3. Organizational Units

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.



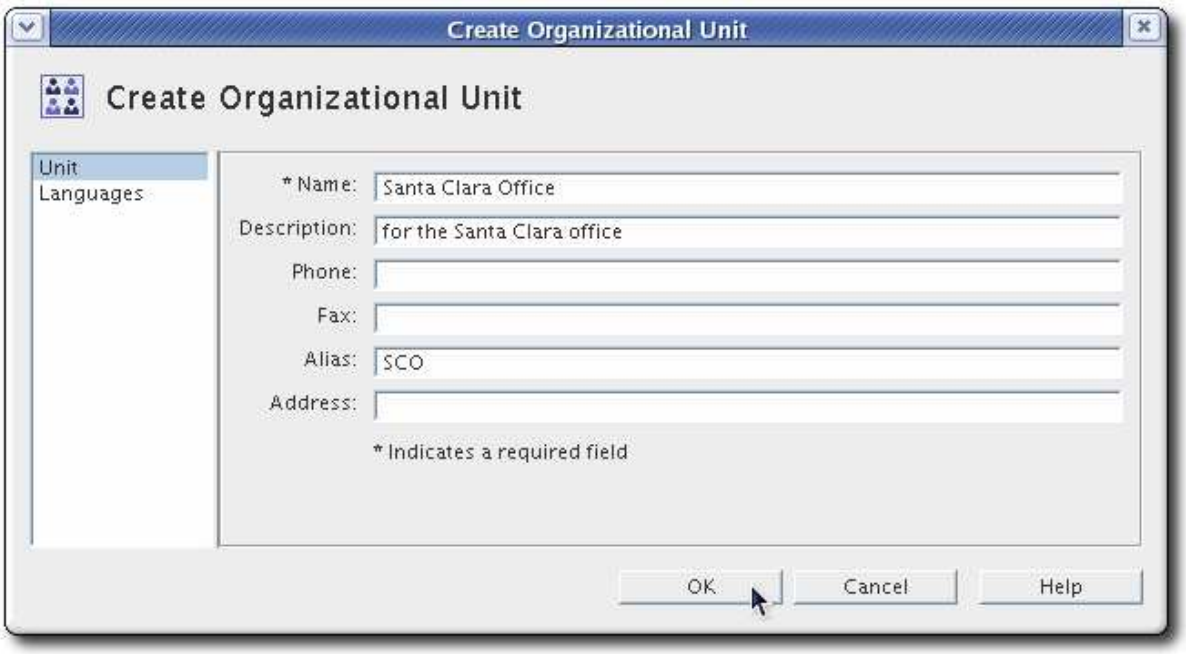
2. Click the **Create** button, and choose **Organizational Unit**.



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**.

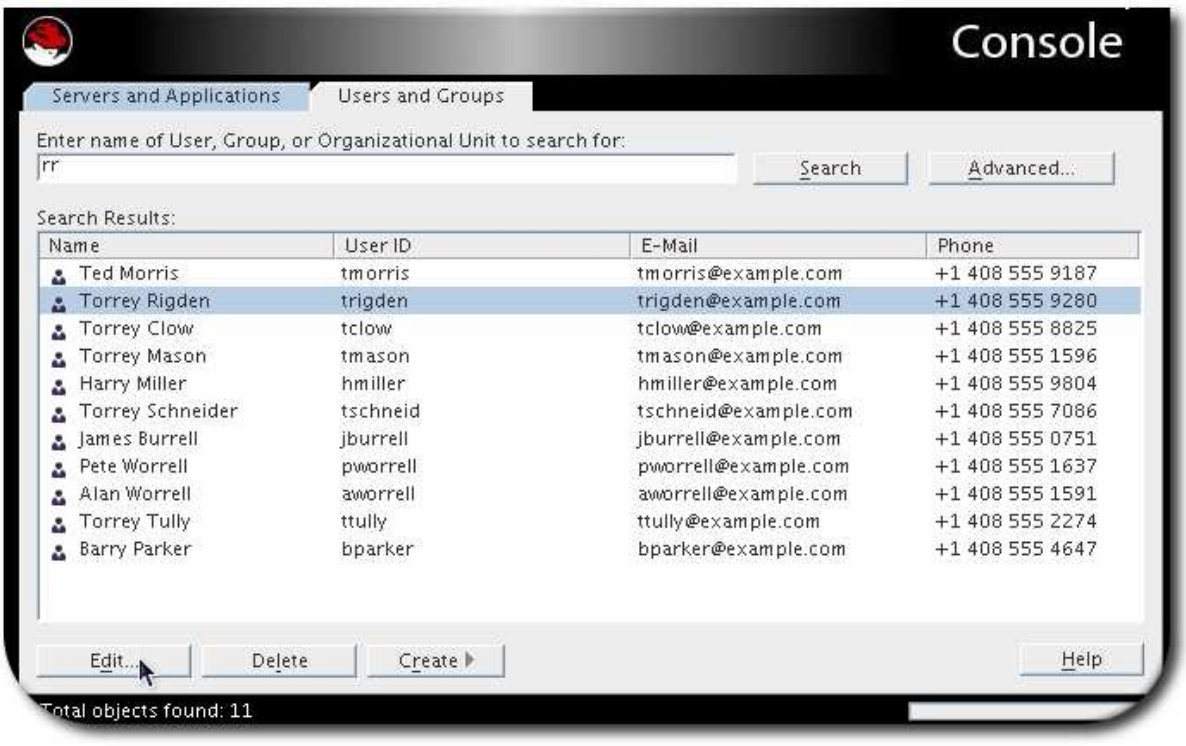
G.4.3. Modifying Directory Entries

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 *Red Hat Directory Server Administration 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 Administration 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 *Administration Server Administrator*, who can perform a limited number of tasks, such as starting, stopping, and restarting servers. The Administration Server Administrator is created so that a user can log into the Red Hat Management Console when the Directory Server is not running. The Administration Server Administrator does not have an LDAP entry; it exists in the Administration Server's configuration file, **/usr/share/dirsrv/properties/admpw**.

IMPORTANT

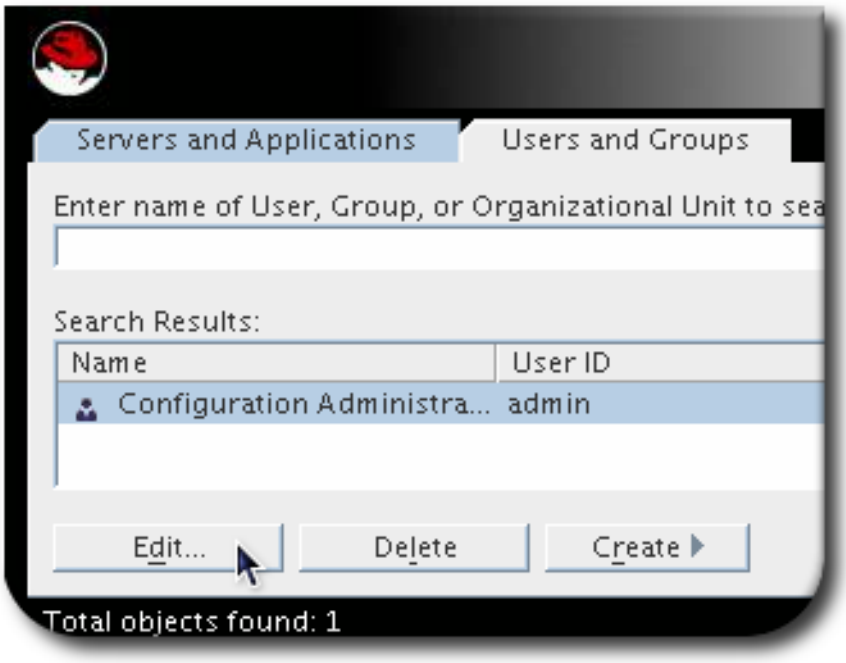
Even though they are created at the same time during installation, and are identical at that time, the Configuration Administrator and Administration Server Administrator are two separate entities. If the user name or password is changed for one, Red Hat Management Console does not automatically make 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”

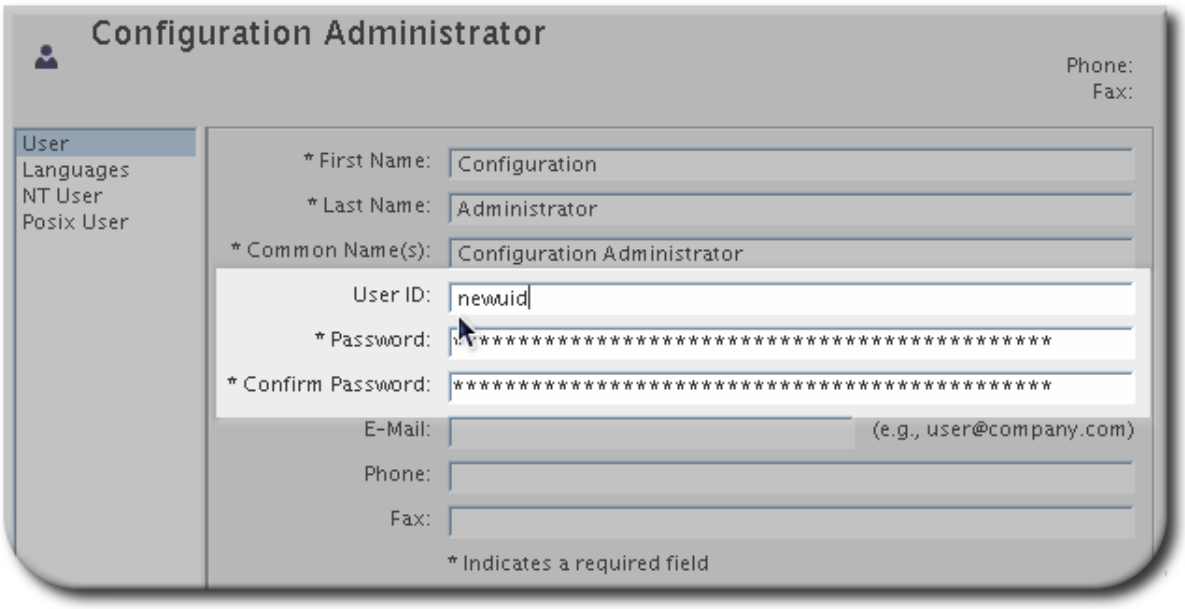
G.4.3.3.1. Changing the Configuration Administrator and Password

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.

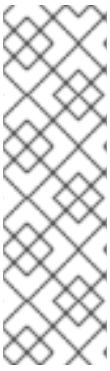
3. Select the Configuration Administrator from the list of search results, and then click **Edit**.



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

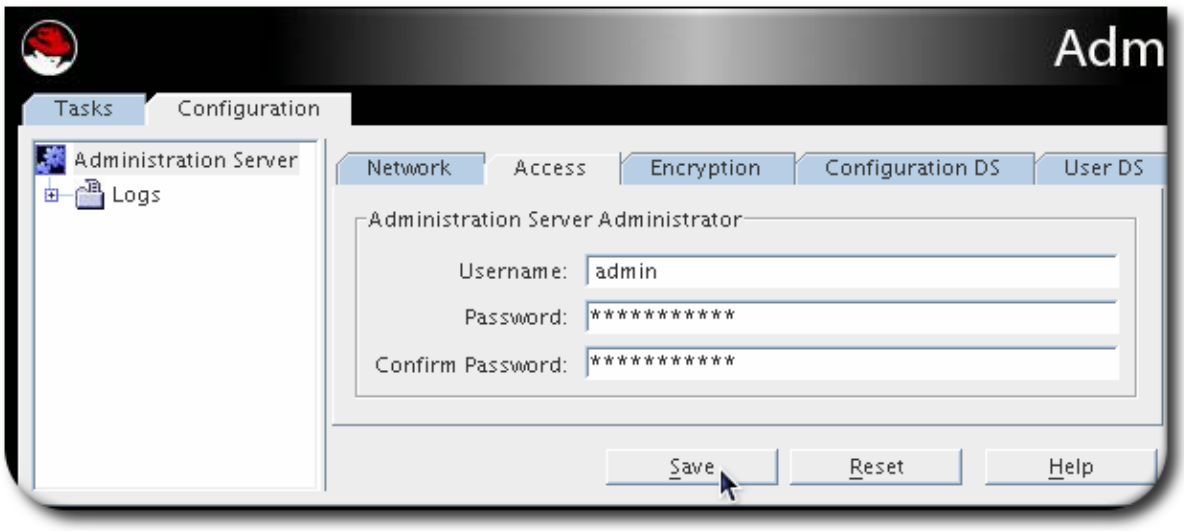
If you are logged into the Console as the Configuration Administrator when you edited the Configuration Administrator entry, update the login information for the directory.

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 Administration 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.

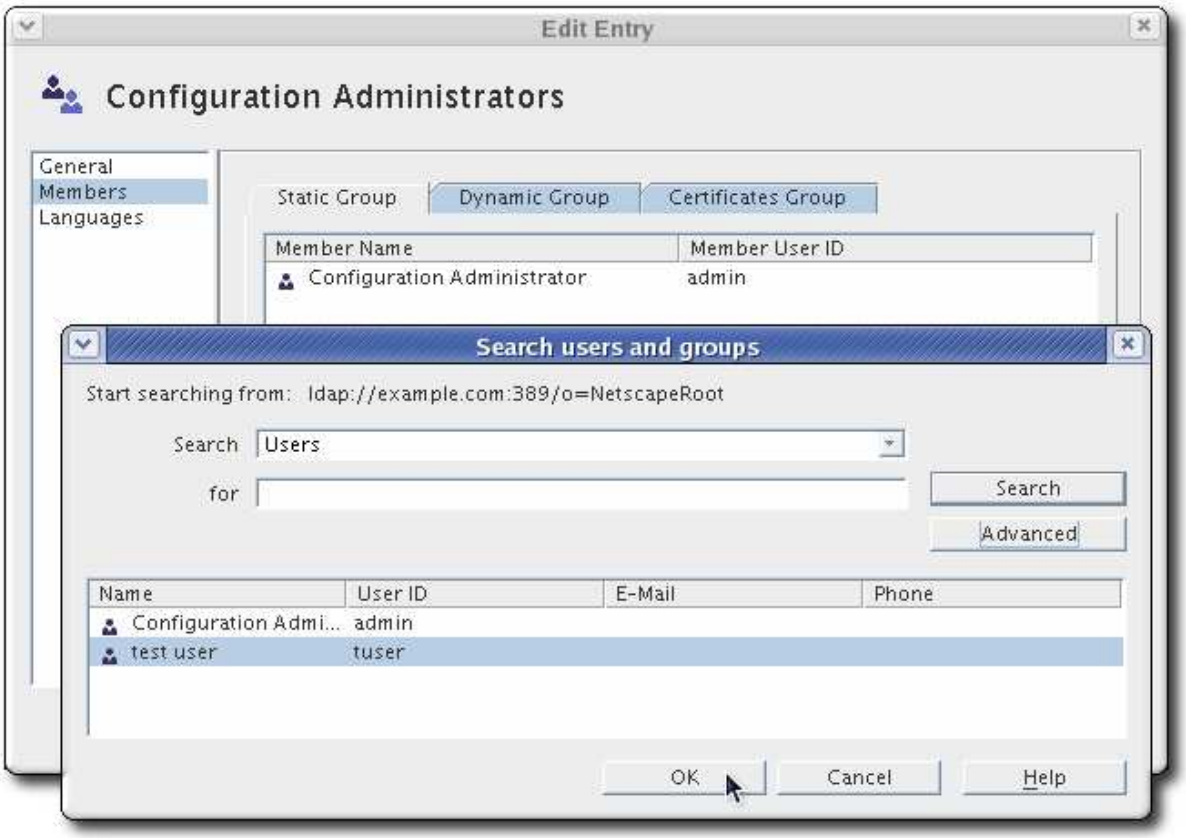
4. Click **Save**.
5. Restart the Administration Server.

```
systemctl restart dirsrv-admin.service
```

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 Administration Server instances managed in the Console.

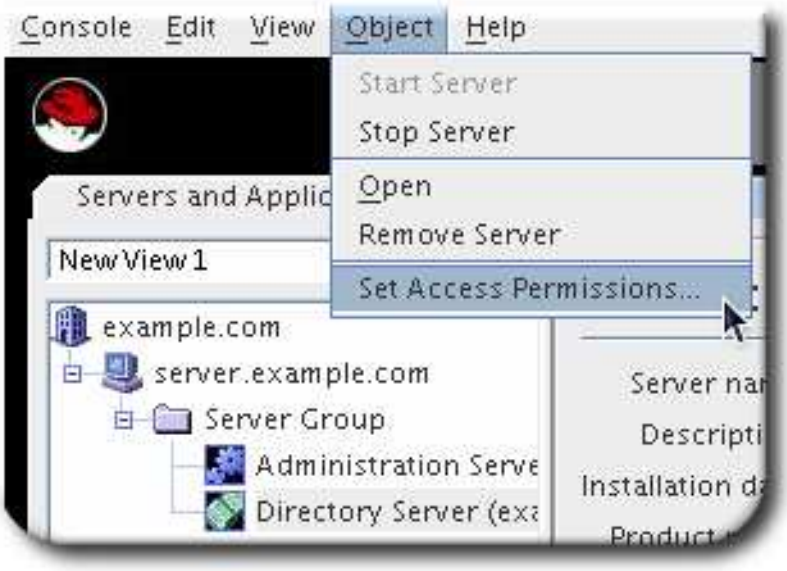
ACIs define what operations users can do with a specific instance of Red Hat Directory Server or Administration Server. ACIs set rules on areas of the subtree which can be accessed or modified, what operations are allowed, even what hosts can be used to access the server and what times of day access 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 Administration Server

Users can be granted administrative privileges, the same as the **admin** user for the Administration 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).

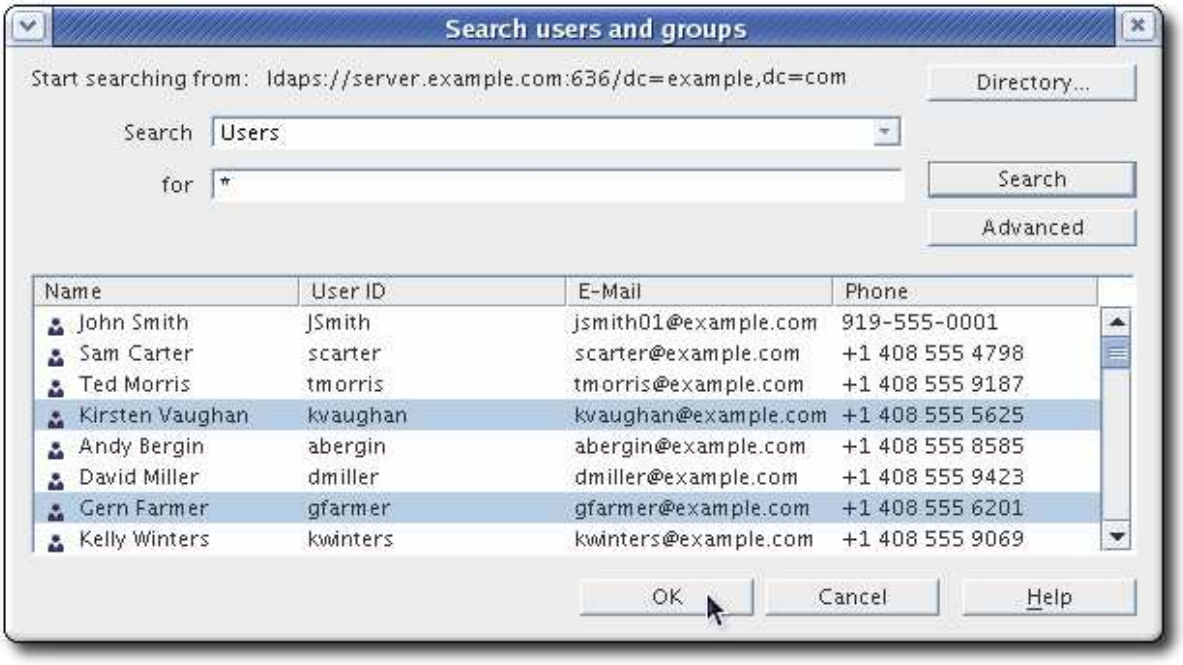
1. Highlight a server in the Console navigation tree.
2. Select the **Object** menu, and choose **Set Access Permissions**.



Alternatively, right-click the entry, and choose **Set Access Permissions**.

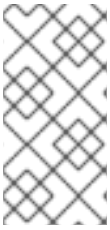
3. Click **Add** to add a new user to the list of administrators for the server. The default users, **Directory Manager** for the Directory Server and **admin** for the Administration Server, are not listed in the **Set Permissions Dialog** box.

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

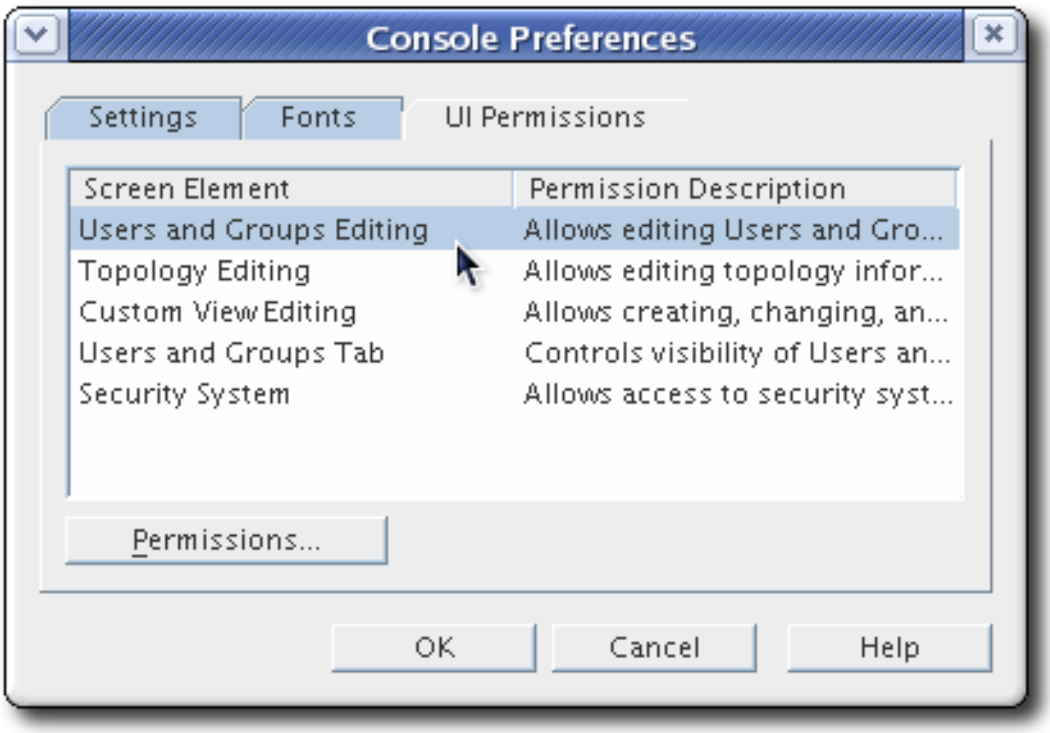
- SIE (host) group permissions

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 Administration Server instances.

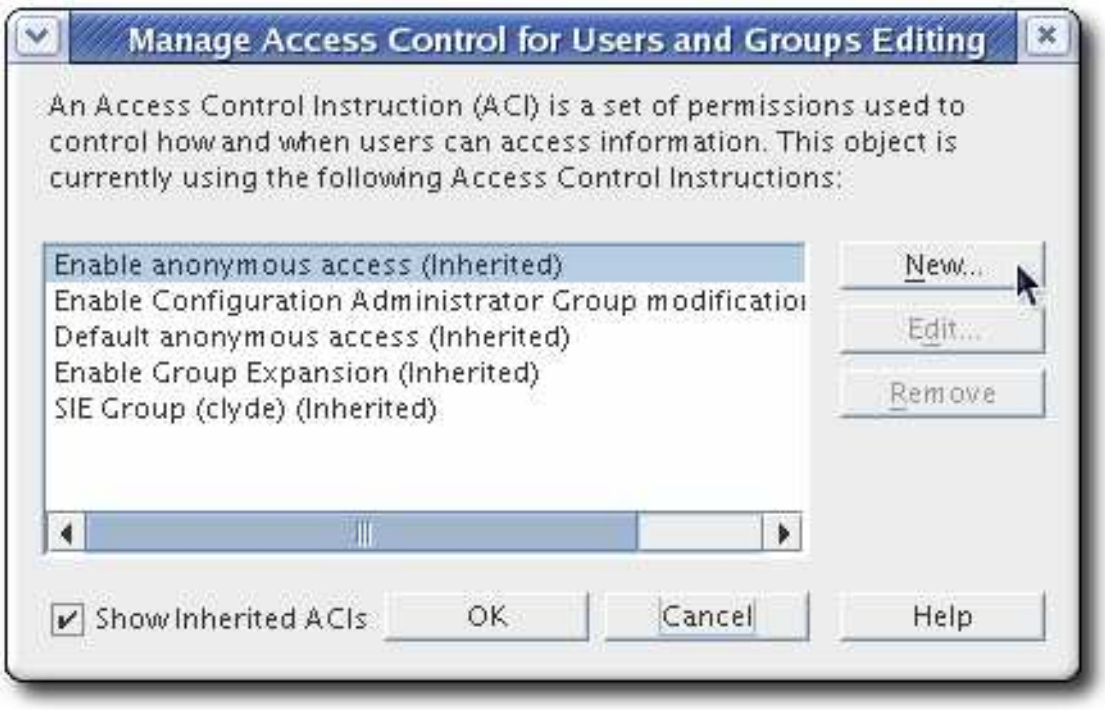
To create new ACIs:

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.

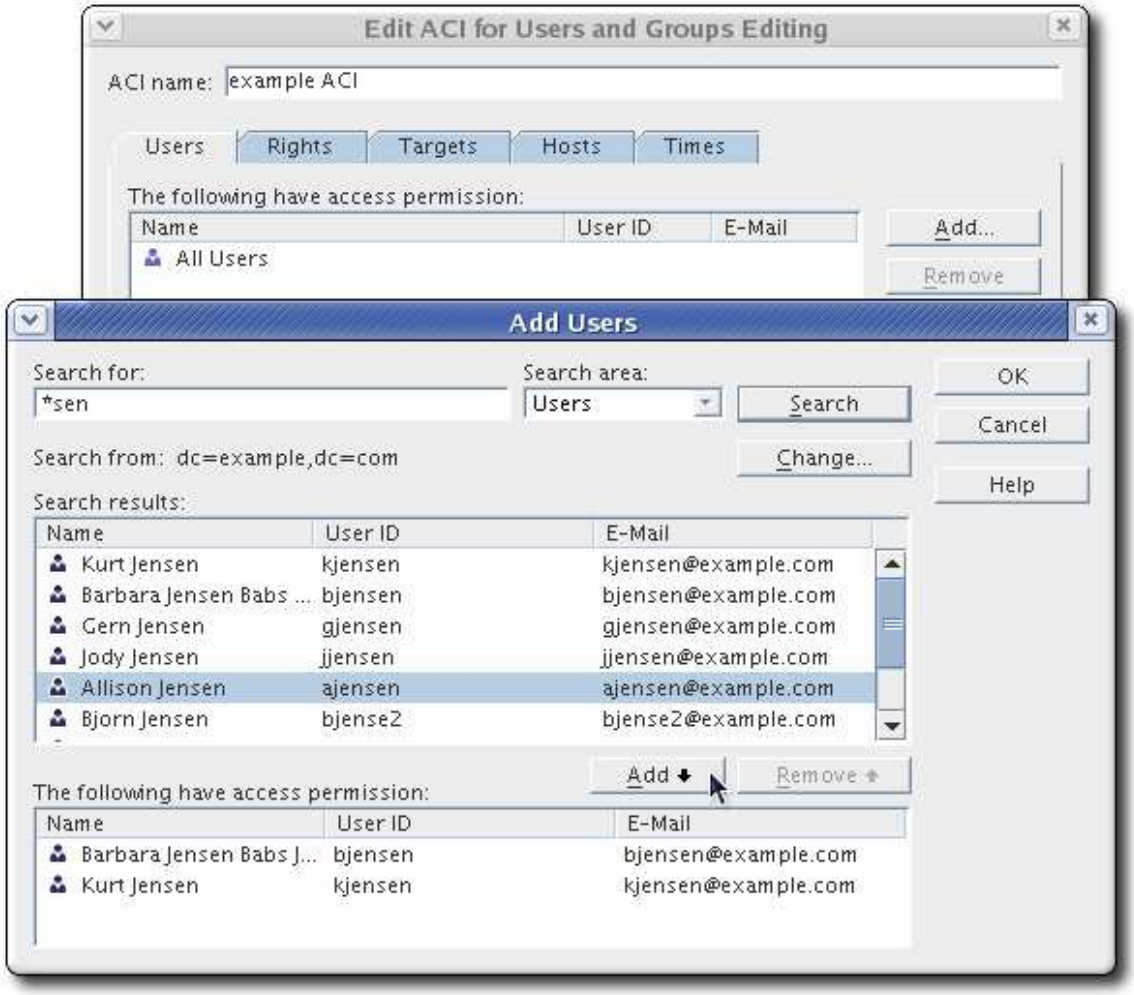


The five inherited ACIs are not displayed by default; to see them listed, click the **Show inherited ACIs** check box.

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.



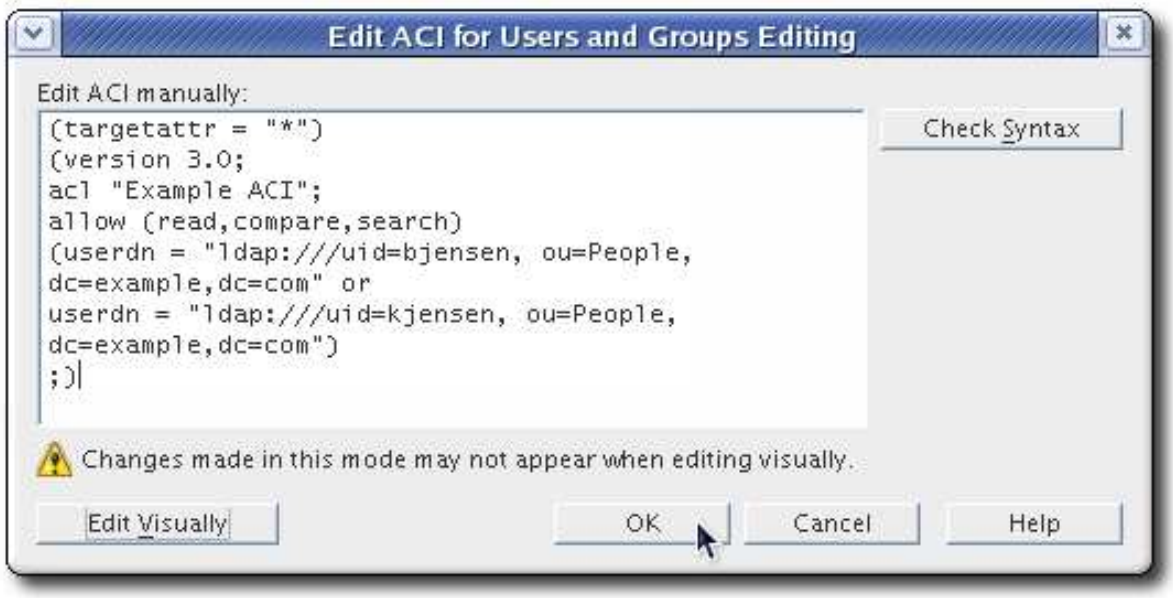
Select the users from the results list and click the **Add** button to include them. Click **OK** to save the list.

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, hostnames, 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.

INDEX

A

B

C

D

E

F

J

K

L

N

O

P

R

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 10.6-2 **Mon Dec 07 2020** **Marc Muehlfeld**
Added a statement that this documentation is deprecated and no longer maintained.

Revision 10.6-1 **Tue Aug 11 2020** **Marc Muehlfeld**
Red Hat Directory Server 10.6 release of the guide.

Revision 10.5-1 **Tue Mar 31 2020** **Marc Muehlfeld**
Red Hat Directory Server 10.5 release of the guide.

Revision 10.4-2 **Mon Aug 26 2019** **Marc Muehlfeld**
Added the *Adding the CA Certificate Used By Directory Server to the Trust Store of Red Hat Enterprise Linux* section.

Revision 10.4-1 **Tue Aug 06 2019** **Marc Muehlfeld**
Red Hat Directory Server 10.4 release of the guide.

Revision 10.3-2 **Wed Jun 05 2019** **Marc Muehlfeld**
Fixed incorrect ACI examples.

Revision 10.3-1 **Wed Oct 24 2018** **Marc Muehlfeld**
Red Hat Directory Server 10.3 release of the guide.

Revision 10.2-1 **Tue Apr 10 2018** **Marc Muehlfeld**
For version 10.2: Added section *Comparing Two Directory Server Instances*, updated *How password policy controls work*, *Solving Common Replication Conflicts*, and *Solving Naming Conflicts*.

Revision 10.1-17 **Mon Mar 12 2018** **Marc Muehlfeld**
Rewrote large parts of the *Managing Access Control* chapter.

Revision 10.1-16 **Wed Feb 14 2018** **Marc Muehlfeld**
Rewrote *Setting up an SNMP Agent for Directory Server*. Added *Generating and Installing a Self-signed Certificate*.

Revision 10.1-15 **Wed Jan 17 2018** **Marc Muehlfeld**
Rewrote *Enabling TLS* section. Added *Configuring Replication Partners to use Certificate-based Authentication* section.

Revision 10.1-14 **Tue Dec 05 2017** **Marc Muehlfeld**
Enhanced the *Managing the NSS Database Used by Directory Server*, *Using Certificate-based Client Authentication* and *Setting Encryption Ciphers* sections.

Revision 10.1-13 **Mon Nov 06 2017** **Marc Muehlfeld**
Moved *Setting Replication Session Hooks* section to the *Plug-in Guide*. Several minor updates.

Revision 10.1-11 **Tue Aug 08 2017** **Marc Muehlfeld**
Rewrote the *Managing Entries Using the Command Line* and *Enforcing Attribute Uniqueness* sections.

Revision 10.1-10 **Tue Aug 01 2017** **Marc Muehlfeld**
For version 10.1.1: Added the *Managing FIPS Mode Support* chapter. Rewrote the *Managing the Directory Manager Password* and *Validating the Syntax of Existing Attribute Values* sections. Multiple minor updates.

Revision 10.1-9 **Mon Jul 31 2017** **Marc Muehlfeld**
Rewrote: *Enabling and Disabling Referential Integrity* section. Updated replication-related screen captures. Multiple minor updates.

Revision 10.1-8 **Wed Jul 12 2017** **Marc Muehlfeld**
Added sections: *Configuring Plug-ins* and *Starting the Directory Server Management Console*. Multiple minor updates.

Revision 10.1-7 **Mon Jun 26 2017** **Marc Muehlfeld**
Rewrote sections: *Trimming the Replication Changelog* and *Moving the Replication Changelog Directory*.

Revision 10.1-6 **Mon May 29 2017** **Marc Muehlfeld**
Updated section: *Setting Access Control for VLV Information*.

Revision 10.1-5 **Tue Mar 14 2017** **Marc Muehlfeld**
Added section: *The Replication Keep-alive Entry*.

Revision 10.1-4 **Fri Feb 24 2017** **Marc Muehlfeld**
Added sections: *Fine Grained ID List Size*, *Trimming the Replication Changelog*, and *Setting up Content Synchronization with an RFC 4533-aware LDAP Servers*. Other minor fixes.

Revision 10.1-3 **Wed Jan 11 2017** **Marc Muehlfeld**
Rewrote section: *Configuring Log Files*. Updated replicate_now.sh example.

Revision 10.1-1 **Fri Nov 11 2016** **Marc Muehlfeld**
Updated supported JRE version. Removed legacy replication support. Other minor fixes.

Revision 10.1-0 **Mon Oct 31 2016** **Marc Muehlfeld**
Red Hat Directory Server 10.1 release of the guide.

Revision 10.0-3 **Wed Jun 22 2016** **Petr Bokoč**
Updated section: *Enabling TLS/SSL in the Directory Server, Administration Server, and Console*. Other minor updates.

Revision 10.0-2 **Thu Mar 24 2016** **Petr Bokoč**
Updated section: *Schema Replication*. Added section: *High-resolution Time Stamps*. Other minor fixes.

Revision 10.0-1 **Wed Jun 17 2015** **Tomáš Čapek**
Updated section: *System Requirements*

Revision 10.0-0 **Tue Jun 09 2015** **Tomáš Čapek**
Red Hat Directory Server 10 release of the guide.