Novell Developer Kit

www.novell.com

LDAP TOOLS

March 1, 2006

Novell®

Legal Notices

Novell, Inc.
404 Wyman Street, Suite 500
Waltham, MA 02451
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

Third-Party Materials

Contents

LDAP Tools

A number of tools have been developed to import entries from a file to an LDAP directory, to modify the entries in a directory from a file, and to export the entries to a file. These tools also support schema modifications by adding attribute and class definitions from a file.

Two types of tools are included in this SDK:

- Novell Import Convert Export utility developed by Novell to perform these tasks as quickly as possible using LDIF files as well as another LDAP server.
- LDAP utilities which are similar to utilities developed by other LDAP SDKs that perform some of these tasks using an LDIF file.

This guide contains the following sections:

- Novell Import Convert Export Utility
- Other LDAP Utilities
- LDIF Examples
- Revision History

Audience

This guide is intended for advanced application developers who are familiar with LDAP Libraries for C, other Novell technologies, and C development environments.

Feedback

We want to hear your comments and suggestions about this manual. Please use the User Comments feature at the bottom of each page of the online documentation and enter your comments there.

Documentation Updates

For the most recent version of the *LDAP Tools Guide*, visit the LDAP Libraries for C Web site (http://developer.novell.com/ndk/cldap_doc.htm) .

Additional Documentation

For the most recent version of NDK guides, see NDK Download Web site (http://developer.novell.com/ndk/downloadaz.htm).

Documentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a backslash. Users of platforms that require a forward slash, such as Linux* or UNIX*, should use forward slashes as required by your software.

Novell Import Convert Export Utility

1

The Novell Import Convert Export tool is a full-featured LDAP utility that enables you to perform:

- LDIF imports
- LDIF exports
- Comma-delimited data imports
- Comma-delimited data exports
- Data migration between LDAP servers
- Schema Compare and Update
- Bulk loading of LDAP data generated from a template

Novell recommends using the Novell Import Convert Export utility because it has the following advantages:

- **Asynchronous Requests.** The LDAP utilities use synchronous processing to ensure that all records are processed in the order they appear in the file. The LDAP server is idle from the time it sends its response to one request until it receives the next request. For a small file, this delay is not significant, but when you are importing a file with a million records, the delay becomes significant. The Novell Import Convert Export utility uses the LDAP Bulk Update/Replication Protocol (LBURP), which allows it to send asynchronous requests that contain more than one update operation while still guaranteeing that the records are processed in order. For more information, see Section 1.8, “LDAP Bulk Update/Replication Protocol,” on page 30.
- **Simple Passwords.** The Novell Import Convert Export utility has an option for eDirectory 8.5 that allows simple passwords to be stored securely in the directory. Storing simple passwords is a quick process whereas generating key pairs is a CPU-intensive process. For a small file, generating the key pairs as the users are added is not significant. However for a large file, this option greatly increases the speed with which the password information can be loaded. This simple password can be used to perform NMAS-enabled and LDAP binds to eDirectory.

  When this password is changed, the public and private key pairs are generated enabling logins using the standard eDirectory password.
- **Server to Server Migration.** The Novell Import Convert Export utility supports the migration of LDAP data from one LDAP server to another LDAP server. Traditional LDAP utilities only support importing data from an LDIF file to an LDAP server. With these utilities, you would have to first export the data to an LDIF file and then import the data to the other server.
- **Forward Referencing.** Since eDirectory enforces referential integrity, container entries must be created before user entries that belong to the container. If you have an LDIF file which has been exported from an LDAP server that does not enforce referential integrity, the records may not be in the right order for eDirectory to process and will thus fail. The Novell Import Convert Export utility has an option that creates a forward reference for such a container. This forward reference is then replaced with a normal entry when the record that adds the container is processed. For more information, see Section 1.9, “Forward References,” on page 31.

You can use the utility as either a command line utility or a graphical utility. The comma-delimited data handler, schema cache, and DirLoad drivers are command line utility. For information about the graphical utility, see “Novell Import Conversion Export Utility” in the *eDirectory Administration Guide* (http://www.novell.com/documentation/eDirectory.html). This reference contains instructions for using the command line utility.

The Novell Import Convert Export is compatible with eDirectory 8.5 or higher. The utility includes two LDAP extension modules that are automatically loaded when the LDAP server starts, and a client utility that runs as a snap-in to ConsoleOne™. The utility replaces both the BULKLOAD and ZoneImport utilities included with previous versions of eDirectory.

The utility is installed as part of ConsoleOne. A Win32 version (`ICE.EXE`), a NetWare version (`ICE.NLM`), Solaris version (`ICE`), and Linux version (`ICE`) are included in the installation.

The utility manages a collection of handlers that read or write data in a variety of formats. Source handlers read data, while destination handlers write data. A single executable module can be both a source and a destination handler. The utility receives data from a source handler, processes the data, then passes the data to a destination handler.

For example, if you want to import LDIF data into an LDAP directory, the utility uses an LDIF source handler to read an LDIF file and an LDAP destination handler to send the data to the LDAP directory server.

See Chapter 3, “LDIF Examples,” on page 55 for more information about LDIF files.

1.1 Syntax

The utility is launched with the following command:

```
ice <general options>
-S[LDIF | LDAP | DELIM | LOAD] <source options>
-D[LDIF | LDAP | DELIM] <destination options>
```

or when using the schema cache:

```
ice -C <general options>
-S[LDIF | LDAP | DELIM] <source options>
-D[LDIF | LDAP | DELIM] <destination options>
```

Note that when performing an update using the schema cache, an LDIF file is not a valid destination.

General options are optional and must come before any source or destination options. The -S (source) and -D (destination) handler sections can be placed in any order. For more information on these options, see

- Section 1.2, “General Options,” on page 10
- Section 1.3, “Source Handler Options,” on page 11
- Section 1.4, “Destination Handler Options,” on page 16
- Section 1.5, “Schema Cache,” on page 18

1.2 General Options

General options affect the overall processing of the utility.

Table 1-1 *General Options*

Option	Description
-C	Specifies that you are using the schema cache to perform schema compare and update. For more information, see Section 1.5, "Schema Cache," on page 18.
-l <log file>	Specifies a filename where output messages (including error messages) are logged. If this option is not used, error message are sent to `ICE.LOG`.
-o	Overwrites an existing log file. If this flag is not set, messages are appended to the log file instead.
-e <ldif error log file>	Specifies a filename where entries that fail are output in LDIF format. This file can be examined, modified to correct the errors, then reapplied to the directory.
-p <URL>	Specifies the location of an XML placement rule to be used by the utility. Placement rules let you change the placement of an entry. For more information, see Section 1.10, "Using XML Rules," on page 32.
-c <URL>	Specifies the location of an XML creation rule to be used by the utility. Creation rules let you supply missing information that might be needed to allow an entry to be created successfully on import. For more information, see Section 1.10, "Using XML Rules," on page 32.
-s <URL>	Specifies the location of an XML schema mapping rule to be used by the utility. You can use a schema mapping rule to map a schema element on a source server to a different but equivalent schema element on a destination server. For more information, see Section 1.10, "Using XML Rules," on page 32.
-b (NetWare Only)	Do not pause for input at the ICE console screen at the end of execution.

1.3 Source Handler Options

The source handler option (-S) determines the source of the import data.Only one of the following can be specified on the command line.

Table 1-2 *Source Handler Options*

Option	Description
-SLDIF	Specifies that the source is an LDIF file. For a list of supported LDIF options, see "LDIF Source Options" on page 12.
-SLDAP	Specifies that the source is an LDAP server. For a list of supported LDAP options, see "LDAP Source Options" on page 12
-SDELIM	Specifies that the source is a comma-delimited data file. For a list of supported DELIM options, see "DELIM Source Options" on page 14.

Option	Description
-SLOAD	Specifies that the source is a DirLoad template. For a list of supported LOAD options, see Section 1.6, “DirLoad Driver,” on page 20.

Specific source handler options are listed in the following sections:

- “LDIF Source Options” on page 12
- “LDAP Source Options” on page 12
- “DELIM Source Options” on page 14
- “LOAD Source Options” on page 15

1.3.1 LDIF Source Options

The LDIF source handler reads data from an LDIF file then sends it to the destination handler.

Table 1-3 *LDIF Source Options*

Option	Description
-f <LDIF file>	Specifies a filename containing LDIF records read by the LDIF source handler and sent to the destination handler.
-a	If the records in the LDIF file are content records (that is, they contain no changetypes), they will be treated as records with a change type of add.
-c	Prevents the LDIF source handler from stopping on errors. This includes errors on parsing LDIF and errors sent back from the destination handler. When this option is set and an error occurs, the LDIF source handler reports the error, finds the next record in the LDIF file, and goes on.
-n	Does not perform update operations, but prints what would be done. When this option is set, the LDIF source handler parses the LDIF file but does not send any records to the destination handler.
-v	Enables the verbose mode of the handler.
-k	Generates the LDIF file in the current system locale.
-g <value>	Language of the LDIF file.

1.3.2 LDAP Source Options

The LDAP source handler reads data from an LDAP server by sending a search request to the server. It then sends the search entries it receives from the search operation to the destination handler.

Table 1-4 *LDAP Source Options*

Option	Description
-s <server name>	Specifies the DNS name or IP address of the LDAP server to which the handler binds and sends a search request. If this option is omitted, it defaults to the local host (127.0.0.1).
-p <port>	Specifies the integer port number of the LDAP server specified by the -s option. If this option is omitted, it defaults to 389.
-d <DN>	Specifies the distinguished name of the entry that should be used when binding to the server. If this option is omitted, the handler performs an anonymous bind.
-w <password>	Specifies the password attribute of the entry specified by the -d option.
-F <search filter>	Specifies an RFC 2254 compliant search filter. If this option is omitted, the search filter defaults to objectclass=*. If the filter includes characters that have special meaning to your command line shell, such as (,), &, and |, you may need to enclose the filter in double quotes. For more information on the format of the search filter, see "LDAP Search Filters" in *LDAP and eDirectory*.
-n	Specifies not to perform the search, but to show what search would be performed.
-a <attribute list>	Specifies a comma-separated list of attributes to retrieve as part of the search. In addition to an attribute list, the option supports the following values: 1.1—Get no attributes *—Get all non-operational attributes An empty list gets all non-operational attributes. Operational attributes must be listed by name. To get all standard attributes and a few operational attributes, use the * with a list of the names of the operational attributes. If this option is omitted, the attribute list defaults to the empty list.
-o <attribute list>	Specifies a comma-separated list of attributes to be omitted from the search results received from the LDAP server before they are sent to the destination handler. This option is very useful in cases where you want to use a wildcard with the -a option to get all attributes of some class and then remove a few of them from the search results before passing the data on to the destination handler. For example, -a* -otelephoneNumber searches for all attributes and then filters the telephoneNumber from the results.
-R	Disables the automatic following of referrals.
-e <value>	Specifies which debugging flags should be enabled in the LDAP client SDK. (See the SDK for supported flags.)

Option	Description
-b <base DN>	Specifies the distinguished name of the entry where the search request begins. If this option is omitted, the base DN defaults to ""(empty string) and the tree root is used as the base DN for the search.
-c <search scope>	Specifies the scope of the search request. The option supports the following values: • One—search only the immediate subordinates of the base object. • Base—search only the base object. • Sub—search the base object and all of its subordinates. If this option is omitted, the search scope defaults to subtree.
-r <deref aliases>	Specifies the way aliases should be dereferenced during the search operation. The option supports the following values: • Never—aliases are never dereferenced when locating the base object or searching. • Always—aliases are always dereferenced when locating the base object and searching. • Search—aliases are dereferenced when searching subordinates of the base object but not when locating the base object. • Find—aliases are dereferenced when locating the base object but not when search the subordinates of the base object. If this option is omitted, the alias dereferencing behavior defaults to never.
-l <time limit>	Specifies, in seconds, the time limit for the search.
-z <size limit>	Specifies the maximum number of entries to be returned by the search.
-V <version>	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
-v	Enables verbose mode of the handler.
-L <filename>	Specifies a file in DER format containing a server key to use for SSL authentication.
-A	Retrieves attribute names only. Attribute values are not returned by the search operation.
-Y <value>	Use SASL authentication (DIGEST-MD5 is the only supported mechanism currently).

1.3.3 DELIM Source Options

The DELIM source handler reads data from a comma-delimited data file then sends it to the destination handler.

Table 1-5 *DELIM Source Options*

Option	Description
-f <filename>	Specifies a filename containing comma-delimited records read by the DELIM source handler and sent to the destination handler.
-F <value>	Specifies a filename containing the attribute data order for the file specified by -f. If this option is not specified, you must enter this information directly using -t. See "Performing a Comma-Delimited Import" on page 29 for more information.
-t <value>	Comma-delimited list of attributes specifying the attribute data order for the file specified by -f. Either this option or -F must be specified. See "Performing a Comma-Delimited Import" on page 29 for more information.
-c	Prevents the DELIM source handler from stopping on errors. This includes errors on parsing comma-delimited data files and errors sent back from the destination handler. When this option is set and an error occurs, the DELIM source handler reports the error, finds the next record in the comma-delimited data file, and goes on.
-n <value>	Specifies the LDAP naming attribute for the new object. This attribute must be contained in the attribute data you specify using -F or -t.
-l <value>	Specifies the path to append the RDN to (such as o=myCompany). If you are passing the DN this value is not necessary.
-o <value>	Comma-delimited list of object classes (if none are contained in your input file), or additional object classes such as auxiliary classes. The default value is inetorgperson.
-i <value>	Comma-delimited list of columns to skip. This value is an integer specifying the number of the column to skip. For example, to skip the third and fifth columns you would pass `-i3,5`.
-d <value>	Specifies the delimiter. The default delimiter is a comma (,). The following values are special case delimeters: [q] = quote (a single " as the delimeter) [t] = tab For example, to specify a tab as a delimiter, you would pass: `-d[t]`
-q <value>	Specifies the secondary delimiter. The default secondary delimiter is a single double quote ("). The following values are special case delimeters: [q] = quote (a single " as the delimeter) [t] = tab For example, to specify a tab as a delimiter, you would pass: `-d[t]`

1.3.4 LOAD Source Options

Table 1-6 *LOAD Source Options*

Option	Description
-f <filename>	Specifies the template file containing all attribute specification and all control information for running the program. This is mandatory. See the DirLoad driver section for details on using this source handler.

Option	Description
-c	Continue to the next record if error is reported.
-v	Verbose messages.
-r	Change the request to a delete request so the data is deleted instead of added. This allows you to remove records that were added using a DirLoad template.
-m	Indicates that modify requests will be in the template file.

1.4 Destination Handler Options

The destination handler options (-D) specify the destination of the export data. Only one of the following can be specified on the command line.

Table 1-7 *Destination Handler Options*

Option	Description
-DLDIF	Specifies that the destination is an LDIF file. For a list of supported options, see "LDIF Destination Options" on page 16
-DLDAP	Specifies that the destination is an LDAP server. For a list of supported options, see "LDAP Destination Options" on page 17.
-DDELIM	Specifies that the destination is a comma-delimited file. For a list of supported options, see "DELIM Destination Options" on page 18.

Specific destination handler options are listed in the following sections:

- "LDIF Destination Options" on page 16
- "LDAP Destination Options" on page 17
- "DELIM Destination Options" on page 18
- "LOAD Destination Options" on page 18

1.4.1 LDIF Destination Options

The LDIF destination handler receives data from the source handler and writes it to an LDIF file.

Table 1-8 *LDIF Destination Options*

Option	Description
-f <LDIF file>	Specifies the filename where LDIF records can be written.
-v	Enables verbose mode of the handler.

Option	Description
-k	Generates the LDIF file in the current system locale.
-g <value>	Language of the LDIF file.

1.4.2 LDAP Destination Options

The LDAP destination handler receives data from the source handler and sends it to an LDAP server in the form of update operations to be performed by the server.

Table 1-9 *LDAP Destination Options*

Option	Description
-s <server name>	Specifies the DNS name or IP address of the LDAP server to which the handler binds and sends a search request. If this option is omitted, it defaults to the local host (127.0.0.1).
-p <port>	Specifies the integer port number of the LDAP server specified by the -s option. If this option is omitted, it defaults to 389.
-d <DN>	Specifies the distinguished name of the entry that should be used when binding to the server.
-w <password>	Specifies the password attribute of the entry specified by the -d option.
-B	Disables the use of asynchronous LBURP requests for transferring update operations to the server. The handler uses standard synchronous LDAP update operation requests instead. See Section 1.8, “LDAP Bulk Update/Replication Protocol,” on page 30 for more information.
-F	Allows the creation of forward references. When an entry is going to be created before its parent exists, a placeholder called a forward reference is created for the entry’s parent to allow the entry to be successfully created. If a later operation creates the parent, the forward reference is changed into a normal entry. For more information, see Section 1.9, “Forward References,” on page 31.
-l	Stores password values using the simple password method of Novell's Modular Authentication Service (NMAS). Passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers. This improves the speed with which an object that has password information can be loaded.
-e <value>	Specifies which debugging flags should be enabled in the LDAP client SDK (see the client SDK for supported values).
-V <version>	Specifies the LDAP protocol version to be used for the connection. It must be 2 or 3. If this option is omitted, the default is 3.
-v	Enables verbose mode of the handler
-L <filename>	Specifies a file in DER format containing a server key to use for SSL authentication.
-Y <value>	Use SASL authentication (DIGEST-MD5 is the only supported mechanism currently).

1.4.3 DELIM Destination Options

The DELIM destination handler receives data from the source handler and writes it to a comma-delimited data file.

Table 1-10 *DELIM Destination Options*

Option	Description
-f <filename>	Specifies the filename where comma-delimited records can be written.
-v	Enables verbose mode of the handler.
-F <value>	Specifies a filename containing the attribute data order for the source data. If this option is not specified, you must enter this information directly using -t.
-t <value>	Comma-delimited list of attributes specifying the attribute data order for the source data. Either this option or -F must be specified.
-l <value>	Can be either RDN or DN. Specifies whether the driver should place the entire DN or just the RDN in the data. RDN is the default value.
-d <value>	Specifies the delimiter. The default delimiter is a comma (','). The following values are special case delimeters: [q] = quote (a single " as the delimeter) [t] = tab For example, to specify a tab as a delimiter, you would pass: `-d[t]`
-q <value>	Specifies the secondary delimiter. The default secondary delimiter is a single double quote ("). The following values are special case delimeters: [q] = quote (a single " as the delimeter) [t] = tab For example, to specify a tab as a delimiter, you would pass: `-d[t]`
-n <value>	Specifies a naming attribute to append during import, such as cn.

1.4.4 LOAD Destination Options

The LOAD handler cannot be used as an destination handler.

1.5 Schema Cache

The Novell Import Convert Export Utility contains a schema cache that enables you to compare and update LDAP schema. The following samples provide details on how to use the schema compare and update.

The schema cache is triggered by passing -C to the ice engine. To perform a comparison, use:

```
ice -C -c[LDIF file for the compare results]
```

To perform an update, use:

```
ice -C -a
```

Compare Example 1

```
ice -C -ccompare.ldif -SLDAP -s192.168.0.1 -p389 -DLDAP -s192.168.0.2
-p389
```

Schema 1: LDAP server 192.168.0.1, port 389
Schema 2: LDAP server 192.168.0.2, port 389
Source Handler: NONE
Destination Handler: LDIF -fcompare.ldif

If no search filter is specified, this will check the Root DSE of both servers to see where their schema is located. Currently if either one has multiple subschema subentries the compare will fail. If a subschema subentry is explicitly specified, it will use that subentry without checking the Root DSE. Compare.ldif will contain a list of LDIF records that if applied to the server 192.168.0.2 would add all of the schema definitions in the schema of the server at 192.168.0.1.

Compare Example 2

```
ice -C -ccompare.ldif -SLDAP -s192.168.0.1 -p389 -DLDIF -fschema2.ldif
```

Schema 1: LDAP server 192.168.0.1, port 389
Schema 2: schema2.ldif
Source Handler: NONE
Destination Handler: LDIF -fcompare.ldif

If no search filter is specified, this will check the Root DSE of the server to see where the schema is located. If there are multiple subschema subentries the compare will fail. If a subschema subentry is explicitly specified, it will use that subentry without checking the Root DSE. Compare.ldif will contain a list of LDIF records that if applied to a server with the schema of schema2.ldif would add all of the schema definitions in the schema of the server at 192.168.0.1.

Compare Example 3

```
ice -C -ccompare.ldif -SLDIF -fschema1.ldif -DLDAP -s192.168.0.2 -p389
```

Schema 1: schema1.ldif
Schema 2: LDAP server 192.168.0.2, port 389
Source Handler: NONE
Destination Handler: LDIF -fcompare.ldif

If no search filter is specified, this will check the Root DSE of the server to see where the schema is located. If there are multiple subschema subentries the compare will fail. If a subschema subentry is explicitly specified, it will use that subentry without checking the Root DSE. Compare.ldif will contain a list of LDIF records that if applied to the server 192.168.0.2 would add all of the schema definitions in the schema of schema1.ldif.

Compare Example 4

```
ice -C -ccompare.ldif -SLDIF -fschema1.ldif -DLDIF -fschema2.ldif
```

Schema 1: schema1.ldif
Schema 2: schema2.ldif
Source Handler: NONE
Destination Handler: LDIF -fcompare.ldif

Compare.ldif will contain a list of LDIF records that if applied to the server with the schema in schema2.ldif would add all of the schema definitions in the schema of schema1.ldif.

Update Example 1

```
ice -C -a -SLDAP -s192.168.0.1 -p389 -DLDAP -s192.168.0.2 -p389
```

Schema 1: LDAP server 192.168.0.1, port 389
Schema 2: LDAP server 192.168.0.2, port 389
Source Handler: NONE
Destination Handler: LDAP server 192.168.0.2, port 389

If no search filter is specified, this will check the Root DSE of both servers to see where their schema is located. Currently if either one has multiple subschema subentries the update will fail. If a subschema subentry is explicitly specified, it will use that subentry without checking the Root DSE. The result will be that the schema of the server at 192.168.0.2 will have all the definitions in the schema at 192.168.0.1.

Note that when performing an update using the schema cache, an LDIF file is not a valid destination.

Update Example 2

```
ice -C -a -SLDIF -fschema1.ldif -DLDAP -s192.168.0.2 -p389
```

Schema 1: schema1.ldif
Schema 2: LDAP server 192.168.0.2, port 389
Source Handler: NONE
Destination Handler: LDAP server 192.168.0.2, port 389

If no search filter is specified, this will check the Root DSE of the server to see where the schema is located. Currently if either one has multiple subschema subentries the update will fail. If a subschema subentry is explicitly specified, it will use that subentry without checking the Root DSE. The result will be that the schema of the server at 192.168.0.2 will have all the definitions in the schema of schema1.ldif.

Note that when performing an update using the schema cache, an LDIF file is not a valid destination.

1.6 DirLoad Driver

The DirLoad driver is a source-only driver that generates eDirectory (or other LDAP directory) data from information and commands contained in a template file. DirLoad can generate random numeric values and insert string values from lists, enabling you to load eDirectory with large amounts of generated data for testing or other purposes. The DirLoad driver can also modify existing attributes and attribute values.

To use the DirLoad driver, run the following from the command line:

```
ice -SLOAD -f<template file> -D<destination driver>
```

Other DirLoad command line options are described in "LOAD Source Options" on page 15, and additional control options are specified within the Template File.

The DirLoad driver can send output to an LDAP server, an LDIF file, or a comma-delimited file using the ICE destination drivers. For more information on ICE destination handlers, see Section 1.4, "Destination Handler Options," on page 16.

The main component of the DirLoad driver is the template file, which is described in the following section.

1.6.1 Template File

The template file contains the object, attributes, commands, and control settings for the DirLoad driver. The components of the template file are described in the following three sections:

- "Object Class and Attributes" on page 21
- "Commands" on page 22
- "Control Settings" on page 24

TIP: As a reference while reading this section, you may wish to open the sample template file, named attrs, contained in the cldapsdk\tools\win32 folder. This file is also included in "Sample Template" on page 27.

1.6.2 Object Class and Attributes

The object class and attributes define the characteristics of the object you wish to add. Within the template file, the object class is specified in the following format:

```
objectclass: <classname>
```

The template file must have an objectclass statement, and only one object class can be defined in a template file. Following the objectclass statement, all mandatory attributes and any optional attributes must be specified in the following format:

```
<attributename>:<attributevalue>
```

For example, the following template would add an inetorgperson named Bob Smith to the ou=development,o=acme container, with the telephone number 1-888-555-1212:

```
objectclass: inetorgperson
dn: cn=bsmith,ou=development,o=acme
sn: Smith
givenname: Bob
telephone number: 1-888-555-1212
```

Multiple attribute values are added by listing each attribute type and value on separate lines in the template file. For example, the following template adds three values to the cn attribute:

```
cn: Robert
cn: Bob
cn: Bobbie
```

Using the LDAP attribute name, any LDAP attribute can be added using this format. If you are unsure of an LDAP name, consult the LDAP class and attribute mappings on the LDAP Group object (http://www.novell.com/documentation/lg/ndsedir86/index.html?taoenu/data/a5bwtyl.html) for your tree in ConsoleOne.

"Commands" on page 22 can be used in place of the attribute value to generate random numbers and insert random strings.

1.6.3 Commands

The power of the DirLoad driver lies within its commands. Commands are used within attribute values to insert:

- Unique numeric values
- Random numeric values
- Random string values from a list
- Previously defined attribute values

Unique Numeric Value

This command inserts a numeric value that is unique for a given object into an attribute value. The syntax is as follows:

```
$C[(<format>)]
```

The optional <format> specifies a printf formatting to apply to the value. To use the default format, "%d", no format is specified and the parenthesis are not used. For information on printf formatting consult a C programming guide.

The starting numeric value is specified with the counter control setting (see "Control Settings" on page 24) and is incremented from that point. For example, if you specified:

```
!COUNTER=100100
workforceID: $C
```

The first object added has an employee ID of 100100, the second 100101, and so on.

This value is incremented after each added object. Therefore, if you use $C multiple times in your template, the value is the same within a single object. For example, if you specify:

```
!COUNTER=100
workforceID= $C
jobCode= $C
```

The first object added has a workforceID of 100 as well as a jobCode of 100, the second object added has a workforceID of 101 as well as a jobCode of 101, and so on.

Random Numeric Value

This command inserts a random numeric value from a specified range into an attribute value. The syntax is as follows:

```
$N(<low>-<high>[,<format>])
```

<low> and <high> specify the lower and upper bounds, respectively, that are used as a random number is generated. They are required.

The optional <format> specifies a printf formatting to apply to the value. To use the default format, "%d", no format is specified. For information on printf formatting consult a C programming guide.

Random String Value from a List

This command inserts a randomly selected string from a specified list into an attribute value. The behavior of this command can be further controlled using the cycle and unicycle control settings (see "Control Settings" on page 24). The syntax is as follows:

```
$R(<filename>[,<format>])
```

The filename specifies a relative or absolute path to a file that contains a list of values, and is required. The values are separated by a newline character.

The optional <format> specifies a printf formatting to apply to the value. To use the default format, "%d", no format is specified and the parenthesis are not used. For information on printf formatting consult a C programming guide.

The following list files are included with this package in the `cldapsdk\tools\win32` folder:

- cities
- company
- domain
- first
- initial
- last
- titles

Using the strings contained in these files with this command you can generate hundreds of thousands of unique objects.

Attribute Value

This command inserts a previously defined attribute value into another attribute value. This command is useful to construct a dn or e-mail address from other attributes that are randomly generated. The syntax is as follows:

```
$A(<attributename>)
```

The attribute name specifies the name of the attribute whose value should be inserted, and is required.

In the following example, the dn is constructed using the first character of the givenname, the first character of the initial, and the entire surname. The e-mail address is constructed using the first character of the givenname, the entire surname, and a static string:

```
givenname: $R(first)
initial: $R(initial)
sn: $R(last)
dn:cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn),ou=ds,ou=dev,o=novell
mail: $A(givenname,%.1s)$A(sn)@acme.com
```

Running the previous code in a template file might generate the following LDIF data:

```
dn:cn=JTSmith,ou=ds,ou=dev,o=novell
objectclass: inetorgperson
mail: JSmith@acme.com
```

```
givenname: Jon
initials: T
sn: Smith
```

It is important to understand that no forward references are allowed. Any attribute whose value you wish to use for the value of another attribute must precede the current attribute in the template file. In the example below, the cn (contained as part of the dn) is constructed from givenname, initial, and sn, so each of these attributes must precede the dn in the template file.

```
givenname: $R(first)
initial: $R(initial)
sn: $R(last)
dn:o=novell,ou=dev,ou=ds,cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn)
```

If the dn preceeds the givenname, initial, or sn, an "invalid attribute reference" error occurs.

NOTE: The dn receives special handling in the LDIF file. Regardless of the location of the dn in the template file, it is written first (as per LDIF syntax) to the LDIF file. All other attributes are written in the order in which they appear.

Table 1-11 *Command Quick Reference*

Command	Description
`$C`	Inserts a unique numeric value, starting from !COUNTER.
`$N(<low>-<high>)`	Inserts a random numeric value from within the specified range.
`$R(filename)`	Inserts a random string value from the specified list.
`$A(attribute name)`	Inserts an attribute value from the specified attribute.

1.6.4 Control Settings

Control settings provide additional control during object creation. All controls have an exclamation point '!' as the first character on the line. The controls can be placed anywhere in the file.

Object Count

This setting determines how many objects are created from the template. The syntax is as follows:

```
!OBJECTCOUNT=<value>
```

Counter

This setting provides the starting value for the unique counter command (see "Commands" on page 22). The syntax is as follows:

```
!COUNTER=<value>
```

Cycle

The CYCLE setting is used to modify the behavior of the random string value from a list command (see "Commands" on page 22). This setting is used to cycle in the following two ways:

- Cycle in Order
- Cycle a Block

Cycle in Order

```
!CYCLE=<listname>
```

When the list specified by <list name> is used, the next value from the list is used rather than randomly selecting a value. Once all values have been consumed in order, the list repeats from the beginning.

Cycle a Block

```
!CYCLE=<listname>,BLOCK=<number>
```

Each value from the list specified by <list name> is to be used <number> times before moving to the next value.

For example, the following control uses each value from the list "ou" 10 times before moving to the next value:

```
!CYCLE=ou,BLOCK=10
```

Unicycle

This control specifies a list of sources that are cycled through, from left to right, enabling you to create guaranteed unique values.

If this control is used, the object count control is used only to limit the number of objects to the maximum number of unique objects that can be created from the lists. In other words, if the lists that are defined by unicycle can produce 15,000 objects, then object count can be used to reduce that number, but not to increase it.

For example, you have a list file called "givenname" which contains the following two values:

```
Doug
Carl
```

You also have a list file called "sn" which contains the following three values:

```
Hoffman
Schultz
Grieger
```

If you add the following control setting:

```
!UNICYCLE=givenname,sn
```

Then add the following attribute definition:

```
cn: $R(givenname) $R(sn)
```

the following six common names are created:

```
cn: Doug Hoffman
cn: Karl Hoffman
cn: Doug Schultz
cn: Karl Schultz
cn: Doug Grieger
cn: Karl Grieger
```

The object count control can be used in conjunction with the previous control setting to limit the number of objects created to less than six, but not more than six objects are created regardless of the value of object count.

Table 1-12 *Control Settings Quick Reference*

Control Setting	Description
`!OBJECTCOUNT=<value>`	Specifies the number of objects to be added from the template.
`!COUNTER=<value>`	Specifies the starting value for the unique numeric value ($C) command.
`!CYCLE`	Specifies an option cycle for the random string value from a list ($R) command, enabling you to cycle in order or cycle a value several times before continuing.
`!UNICYCLE`	Specifies lists to cycle in order to generate unique values.

1.6.5 Modify

In addition to adding objects, the DirLoad driver can modify objects by deleting, adding, and replacing attributes or their values. To specify a modify, include the -m option on the command line:

```
ice -SLOAD -m -f<template file> -D<destination driver>
```

When making a modification, the first line of your template file is the dn of the object to be modified.

Delete

This command deletes an attribute and its attribute value. The syntax is as follows:

```
delete: <attributename>
```

NOTE: Mandatory attributes cannot be deleted, however, you can use the replace command to edit the value of a mandatory attribute.

For example, the following template deletes the otherPhoneNumber attribute from the user Gillespie Zammitti:

```
dn: cn=gzammitti,ou=development,o=acme
delete: otherPhoneNumber
```

Add

This command adds an attribute and an attribute value. The syntax is as follows:

```
add: <attributename>
<attributename>: <attributevalue>
```

For example, the following template adds a personalTitle to the user Gillespie Zammitti:

```
dn: cn=gzammitti,ou=development,o=acme
add: personalTitle
personalTitle: Best Programmer in the World
```

Replace

This command replaces an attribute value with a new value. The syntax is as follows:

```
replace: <attributename>
<attributename>: <attributevalue>
```

For example, the following template changes the personalTitle of the user Gillespie Zammitti:

```
dn: cn=gzammitti,ou=development,o=acme
replace: personalTitle
personalTitle: Best Exaggerator in the World
```

Multi-valued attributes can be added with the replace command.

This command replaces all values of an attribute with the values specified in your template. If you wish to keep one or more of the current attribute values use the add command instead.

1.6.6 Sample Template

This section demonstrates the main functionality of the DirLoad driver, and demonstrates using the DirLoad driver with various destination drivers.

These samples use the sample template file, attrs, contained in the `cldapsdk\tools\win32` folder). This template file is as follows:

```
#======================================================================
=
#  DirLoad 1.00
#======================================================================
=
!COUNTER=300
!OBJECTCOUNT=2
#----------------------------------------------------------------------
-
#  ATTRIBUTE TEMPLATE
#----------------------------------------------------------------------
-
objectclass: inetorgperson
givenname: $R(first)
initials: $R(initial)
sn: $R(last)
dn:
```

```
cn=$A(givenname,%.1s)$A(initial,%.1s)$A(sn),ou=$R(ou),ou=dev,o=novell,
telephonenumber: 1-800-$N(1-999,%03d)-$C(%04d)
title: $R(titles)
```

Given the previous template file, running the following command produces the LDIF file displayed below.

```
ice -SLOAD -fattrs -DLDIF -fnew.ldf
```

```
version: 1

dn:cn=JTSmith,ou=ds,ou=dev,o=novell
changetype: add
objectclass: inetorgperson
givenname: Jon
initials: T
sn: Smith
telephonenumber: 1-800-290-0300
title: engineer

dn:cn=TTSmith,ou=ds,ou=dev,o=novell
changetype: add
objectclass: inetorgperson
givenname: Timo
initials: T
sn: Smith
telephonenumber: 1-800-486-0301
title: manager
```

Running the following command sends the data to an LDAP server:

```
ice -SLOAD -fattrs -DLDAP -swww.acme.com -dcn=admin,o=acme -
wpassword
```

Modify

The following sample demonstrates a modify command. If the template file is as follows:

```
#======================================================================
=
#  DirLoad 1.00
#======================================================================
=
!COUNTER=300
!OBJECTCOUNT=1
#----------------------------------------------------------------------
-
#  ATTRIBUTE TEMPLATE
#----------------------------------------------------------------------
-
dn: cn=gzammitti,ou=development,o=acme
add: givenname
givenname: test1
delete: mail
replace: givenname
```

```
givenname: test2
givenname: test3
```

Running the following command produces the LDIF file displayed below:

```
ice -SLOAD -fattrs -m -DLDIF -fnew.ldf
```

```
version: 1

dn: cn=gzammitti,ou=development,o=acme
changetype: modify
delete: mail
-
add: givenname
givenname: test1
-
replace: givenname
givenname: test2
givenname: test3
```

1.7 Sample Commands

Listed below are sample commands that can be used with the utility for the following functions:

- "Performing an LDIF Import" on page 29
- "Performing an LDIF Export" on page 29
- "Performing a Data Migration between LDAP Servers" on page 30

1.7.1 Performing an LDIF Import

To perform an LDIF import, use a command similar to the following:

```
ice -SLDIF -fc:\temp\chapter2.ldif -DLDAP -sserver1.acme.com -p389 -
dcn=admin,c=us -wsecret
```

This particular command line reads LDIF data from the `c:\temp\chapter2.ldif` file and sends it to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us, and password "secret."

1.7.2 Performing an LDIF Export

To perform an LDIF export, use a command similar to the following:

```
ice -SLDAP -sserver1.acme.com -p389 -dcn=admin,c=us -wpassword -
lobjectClass=* -csub -DLDIF -fc:\temp\server1.ldif
```

This command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us, and password "password" and outputs the data in LDIF format to the `c:\tmp\server1.ldif` file.

1.7.3 Performing a Comma-Delimited Import

To perform a comma-delimited import, use a command similar to the following:

```
ice -SDELIM -fc:\tmp\in.csv -Fc:\tmp\order.csv -ncn
-lo=acme -DLDAP -sserver1.acme.com -p389
-dcn=admin,c=us -wsecret
```

This command reads comma-delimited values from the `c:\tmp\in.csv` file and reads the attribute order from the `c:\tmp\order.csv` file. For each attribute entry in `in.csv`, the attribute type is specified in `order.csv`. For example, if `in.csv` contains the following:

```
pat,engineer,555-1212,pat@acme.com,"Acme, Inc."
```

Then order.csv would contain the following:

```
cn,title,phonenumber,emailaddress,company
```

The information in order.csv could be input directly using the -t option.

The data is then sent to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us, and password “secret.”

We specified that cn should become the new DN for this object using the -n option, and we added this object to the organization container acme using the -l option.

1.7.4 Performing a Comma-Delimited Export

To perform a comma-delimited export, use a command similar to the following:

```
ice -SLDAP -sserver1.acme.com -p389 -dcn=admin,c=us -wpassword -
lobjectClass=* -csub -DDELIM
-fc:\tmp\server1.csv -Forder.csv
```

This command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us, and password “password” and outputs the data in comma-delimited format to the `c:\tmp\server1.csv` file.

1.7.5 Performing a Data Migration between LDAP Servers

To perform a data migration between LDAP servers, use a command similar to the following:

```
ice -SLDAP -sserver1.acme.com -p389 -dcn=admin,c=us -wpassword -
lobjectClass=* -csub -DLDAP -sserver2.acme.com -p389 -dcn=admin,c=us -
wpassword
```

This particular command line performs a subtree search for all objects in the server server1.acme.com at port 389 using the identity cn=admin,c=us with password “password” and sends it to the LDAP server server1.acme.com at port 389 using the identity cn=admin,c=us and password “secret.”

1.8 LDAP Bulk Update/Replication Protocol

The utility uses the LDAP Bulk Update/Replication Protocol (LBURP) to send asynchronous requests to an LDAP server. This guarantees that the requests are processed in the order specified by the protocol and not in an arbitrary order influenced by multiprocessor interactions or the operating system’s scheduler.

LBURP also lets the utility send several update operations in a single request and receive the response for all of those update operations in a single response. This adds to the network efficiency of the protocol.

LBURP works as follows:

1. The client utility binds to an LDAP server.
2. The server sends a bind response to the client.
3. The client sends a start LBURP extended request to the server.
4. The server sends a start LBURP extended response to the client.
5. The client sends zero or more LBURP operation extended requests to the server.

   These requests can be sent asynchronously. Each request contains a sequence number identifying the order of this request relative to other requests sent by the client over the same connection. Each request also contains one or more LDAP update operations.
6. The server processes each of the LBURP operation extended requests in the order specified by the sequence number and sends an LBURP operation extended response for each request.
7. After all of the updates have been sent to the server, the client sends an end LBURP extended request to the server.
8. The server sends an end LBURP extended response to the client.

The LBURP protocol lets the utility present data to the server as fast as the network connection between the two allows. This lets the server stay busy processing update operations 100 percent of the time because it never has to wait for the utility to give it more work to do.

The LBURP processor in eDirectory also commits update operations to the database in groups to gain further efficiency in processing the update operations. LBURP can greatly improve the efficiency of your LDIF imports over a traditional synchronous approach.

To disable LBURP during an LDIF import, use the -B option with the destination handler.

IMPORTANT: Since LBURP is a relatively new protocol, eDirectory servers prior to version 8.5 (and most non-eDirectory servers) do not support it. If you are using the Novell Import Conversion and Export utility to import an LDIF file to one of these servers, you must disable the LBURP option for the LDIF import to work.

1.9 Forward References

In an LDIF file, the record to add an entry can come before the record to add its parent container. In eDirectory which guarantees referential integrity, this situation generates an error because the entry's parent does not exist. The -F option for the LDAP destination handler solves this problem. It enables the creation of a forward reference for the parent container. When the record to create the parent is processed, the forward reference is replaced with a normal entry.

It is possible, that after the entire LDIF file is processed, that a few forward references will remain because the LDIF file did not contain records to add them. Such forward references remain in the directory as unknown objects, and the entries below them remain and function as normal entries. You can either add these remaining forward references as entries or move the subordinate entries to another container.

To identify the unknown objects in your directory, you can

- Use ConsoleOne where unknown objects are represented by a round yellow icon with a question mark in the center.
- Use an LDAP search with the search filter set to objectClass=unknown.

Both of these methods display all entries that have an objectClass of unknown, not just the entries that are forward references. From these entries, you need to select the entries to add.

- When eDirectory processes an add for an unknown object that already exists as a forward reference, eDirectory transforms the existing forward reference entry into a normal entry.
- When eDirectory processes an add for an unknown object that isn't a forward reference, eDirectory returns an error of "object already exists."

1.10 Using XML Rules

The Novell Import Convert Export utility allows you to specify a set of rules that are applied to each source record before the record is sent to the destination. These rules are specified in XML and solve the following problems when importing entries from one LDAP directory to another:

- Schema differences—two LDAP directories will seldom, if ever, have identical schemas. Schema mapping rules allow you to map an attribute or class definition in the source directory to an attribute or class definition in the destination directory. For example, since most eDirectory directories use User class for users, you would probably want to map the inetOrgPerson class from the source directory to the User class in the eDirectory destination directory. If the destination schema has no equivalent class or attribute, the schema can be extended to include new classes and attributes. (For more information, see Section 3.7, "Modifying the Schema," on page 62.) For information on the format of these rules, see "Schema Mapping Rules" on page 33.
- Hierarchical differences—two LDAP directories will seldom, if ever, have the same parent containers. As you move entries from one directory to another, you may not want the entries to be created in the same containers in the destination directory. Placement rules allow you to take entries from the source container and place these entries in another container in the destination directory. For example, if two departments (sales1 and sales2) have been combined to one department (sales), you can create a rule that places all the entries from sales1 and sales2 containers in the sales container on the destination server. For information on the format of these rules, see "Placement Rules" on page 36.
- Missing information—two LDAP directories will seldom, if ever, have the same rules for creating new entries. For example, if you are importing entries from a directory that only requires a value for the cn attribute for users to a directory that requires a value for both the cn and the sn attributes, the adds for records which do not have a sn value will fail. Create rules allow you to supply default values for such attributes so that the adds will not fail. For information on the format of these rules, see "Create Rules" on page 34.

You enable rule processing with the -p, -c, and -s general options (see Section 1.2, "General Options," on page 10). Each of these options must be followed by a URL with the following format:

```
file://<[path/]filename>
```

The file must be on the local file system.

The rules use the same XML format as the DirXML product.

1.10.1 Schema Mapping Rules

The <attr-name-map> element is the top level element for the schema mapping rules. Mapping rules determine how the import schema interacts with the export schema. They associate specified import class definitions and attributes with corresponding definitions in the export schema.

Mapping rules can be set up for attribute names or class names.

- For an attribute mapping, the rule must specify that it is an attribute mapping, a name space (nds-name is the tag for the source name), the name in the NDS name space, and then the other name space (app-name is the tag for the destination name) and the name in that name space. It can specify that the mapping applies to a specific class, or it can be applied to all classes with the attribute.
- For a class mapping, the rule must specify that it is a class mapping rule, a name space (eDirectory or the application), the name in that name space, and then the other name space and the name in that name space.

The following is the formal DTD definition of schema mapping rules.

```
<!ELEMENT attr-name-map (attr-name | class-name)*>

<!ELEMENT attr-name (nds-name, app-name)>
<!ATTLIST attr-name
          class-name    CDATA    #IMPLIED>

<!ELEMENT class-name (nds-name, app-name)>

<!ELEMENT nds-name (#PCDATA)>

<!ELEMENT app-name (#PCDATA)>
```

You can have multiple mapping elements in the file. Each element is processed in the order that it appears in the file. If you map the same class or attribute more than once, the first mapping takes precedence.

The following samples illustrate how to create a schema mapping rule.

Schema Rule 1. The following rule maps the source's surname attribute to the destination's sn attribute for the inetOrgPerson class.

```
<attr-name-map>
   <attr-name class-name="inetOrgPperson">
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
</attr-name-map>
```

Schema Rule 2. The following rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
   </class-name>
</attr-name-map>
```

Schema Rule 3. The following example contains two rules. The first rule maps the source's Surname attribute to the destination's sn attribute for all classes that use these attributes. The second rule maps the source's inetOrgPerson class definition to the destination's User class definition.

```
<attr-name-map>
   <attr-name>
      <nds-name>surname</nds-name>
      <app-name>sn</app-name>
   </attr-name>
   <class-name>
      <nds-name>inetOrgPerson</nds-name>
      <app-name>User</app-name>
   </class-name>
</attr-name-map>
```

Sample Command. If the schema rules are saved to an `sr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `outt1.ldf`.

```
ice -o -sfile://sr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

1.10.2 Create Rules

Create rules specify the conditions for creating a new entry in the destination directory. They support the following elements:

- **Required attributes.** Specifies that an add record must have values for all of the required attributes, or the add fails. The rule can supply a default value for a required attribute. If a record does not have a value for the attribute, the entry is given the default value. If the record has a value, the record value is used.
- **Matching attributes.** Specifies that an add record must have the specific attributes and match the specified values, or the add fails.
- **Templates.** Specifies the distinguished name of a template object in eDirectory. The Novell Import Convert Export utility does not currently support specifying templates in create rules.

The following is the formal DTD definition for create rules:

```
<!ELEMENT create-rules (create-rule)*>

<!ELEMENT create-rule (match-attr*,
                       required-attr*,
                       template?) >
<!ATTLIST create-rule
          class-name    CDATA   #IMPLIED
          description   CDATA   #IMPLIED>

<!ELEMENT match-attr    (value)+ >
<!ATTLIST match-attr
          attr-name      CDATA    #REQUIRED>
```

```
<!ELEMENT required-attr (value)*>
<!ATTLIST required-attr
          attr-name     CDATA   #REQUIRED>

<!ELEMENT template EMPTY>
<!ATTLIST template
          template-dn   CDATA   #REQUIRED>
```

You can have multiple create rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped and the skipping does not generate an error.

The following examples illustrate how to format create rules.

Create Rule 1. The following rule places three conditions on add records that belong to the inetOrgPerson class. These records must have givenName and surname attributes. They ought to have an L attribute, but if they don't, the create rule supplies a default value of Provo for them.

```
<create-rules>
   <create-rule class-name="inetOrgPerson">
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 2. The following create rule places three conditions on all add records, regardless of their base class:

- The record must contain a givenName attribute. If it doesn't, the add fails.
- The record must contain a surname attribute. If it doesn't, the add fails.
- The record must contain an L attribute. If it doesn't, the attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
      <required-attr attr-name="givenName"/>
      <required-attr attr-name="Surname"/>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Create Rule 3. The following create rule places two conditions on all records, regardless of base class:

- It checks to see if the record has a uid attribute with a value of ratuid. If it doesn't, the add fails.
- It checks to see if the record has an L attribute. If it does not have this attribute, the L attribute is set to a value of Provo.

```
<create-rules>
   <create-rule>
```

```
      <match-attr attr-name="uid">
         <value>cn=ratuid</value>
      </match-attr>
      <required-attr attr-name="L">
         <value>Provo</value>
      </required-attr>
   </create-rule>
</create-rules>
```

Sample Command. If the create rules are saved to an `cr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `outt1.ldf.`

```
ice -o -cfile://cr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

1.10.3 Placement Rules

Placement rules determine where an entry is created in the destination directory. They support the following conditions for determining whether the rule should be used to place an entry:

- **Match class.** If the rule contains any match class elements, an objectClass specified in the record must match the class-name attribute in the rule. If the match fails, the placement rule is not used for that record.
- **Match attribute.** If the rule contains any match attribute elements, the record must contain an attribute value for each of the attributes specified in the match attribute element. If the match fails, the placement rule is not used for that record.
- **Match path.** If the rule contains any match path elements, a portion of the record's dn must match the prefix specified in the match path element. If the match fails, the placement rule is not used for that record.

The last element in the rule specifies where to place the entry. The placement rule can use zero or more of the following:

- **PCDATA.** Uses parsed character data to specify the DN of a container for the entries.
- **Copy the name.** Specifies that the RDN of the old DN is used in the entry's new DN.
- **Copy the attribute.** Specifies the naming attribute to use in the entry's new DN. The specified naming attribute must be a valid naming attribute for the entry's base class.
- **Copy the path.** Specifies that the source DN should be used as the destination DN.
- **Copy the path suffix.** Specifies that the source DN, or a portion of its path, should be used as the destination DN. If a match-path element is specified, only the part of the old DN that matches the prefix attribute of the match-path element is used as part of the entry's DN.

The following is the formal DTD definition for the placement rule.

```
<!ELEMENT placement-rules (placement-rule*)>
<!ATTLIST placement-rules
          src-dn-format    (%dn-format;)   "slash"
          dest-dn-format   (%dn-format;)   "slash"
          src-dn-delims    CDATA           #IMPLIED
          dest-dn-delims   CDATA           #IMPLIED>
```

```
<!ELEMENT placement-rule (match-class*,
                          match-path*,
                          match-attr*,
                          placement)>
<!ATTLIST placement-rule
          description      CDATA            #IMPLIED>

<!ELEMENT match-class      EMPTY>
<!ATTLIST match-class
          class-name       CDATA            #REQUIRED>

<!ELEMENT match-path       EMPTY>
<!ATTLIST match-path
          prefix           CDATA            #REQUIRED>

<!ELEMENT match-attr       (value)+ >
<!ATTLIST match-attr
          attr-name        CDATA            #REQUIRED>

<!ELEMENT placement        (#PCDATA |
                           copy-name |
                           copy-attr |
                           copy-path |
                           copy-path-suffix)* >
```

You can have multiple placement-rule elements in the file. Each rule is processed in the order that it appears in the file. If a record does not match any of the rules, that record is skipped, and the skipping does not generate an error.

The following examples illustrate how to format placement rules. The scr-dn-format="ldap" and dest-dn-format="ldap" attributes set the rule so that the name space for the dn in the source and destination is LDAP format.

NOTE: The Novell Import Convert Export utility only supports source and destination names in LDAP format.

Placement Example 1. The following placement rule requires that the record have a base class of inetOrgPerson. If the record matches this condition, the entry is placed immediately subordinate to the test container and left-most component of its source dn is used as part of its dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-class class-name="inetOrgPerson"></match-class>
       <placement>cn=<copy-name/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with a base class of inetOrgPerson and with the following dn

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ

would have the following dn in the destination directory

```
dn: cn=Kim Jones, o=test
```

Placement Example 2. The following placement rule requires that the record have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and left-most component of its source dn is used as part of its dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement>cn=<copy-name/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones

would have the following dn in the destination directory

```
   dn: cn=Kim Jones, o=test
```

Placement Example 3. The following placement rule require the record to have an sn attribute. If the record matches this condition, the entry is placed immediately subordinate to the test container and its sn attribute is used as part of its dn. The specified attribute in the copy-attr element must be a naming attribute of the entry's base class.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
  <placement-rule>
    <match-attr attr-name="sn"></match-attr>
    <placement>cn=<copy-attr attr-name="sn"/>,o=test</placement>
  </placement-rule>
</placement-rules>
```

With this rule, a record with the following dn and sn attribute

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones

would have the following dn in the destination directory

```
   dn: cn=Jones, o=test
```

Placement Sample 4. The following placement rule requires the record to have an sn attribute. If the record matches this condition, the source dn is used as the destination dn.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement><copy-path/></placement>
    </placement-rule>
 </placement-rules>
```

Placement Sample 5. The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry's entire DN is copied to the test container.

```
<placement-rules src-dn-format="ldap" dest-dn-format="ldap">
    <placement-rule>
       <match-attr attr-name="sn"></match-attr>
       <placement><copy-path-suffix/>,o=test</placement>
    </placement-rule>
 </placement-rules>
```

With this rule, a record with the following dn and sn attribute

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones

would have the following dn in the destination directory

```
   dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ, o=test
```

Sample Command. If the placement rules are saved to an `pr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `foutt1.ldf`.

```
ice -o -pfile://pr1.xml -SLDIF -f1entry.ldf -c -DLDIF
-foutt1.ldf
```

Placement Sample 6: The following placement rule requires the record to have an sn attribute. If the record matches this condition, the entry's entire DN is copied to the neworg container.

```
<placement-rules> <placement-rule>  <match-path
prefix="o=engineering"/>  <placement><copy-path-suffix/>o=neworg</
placement> </placement-rule></placement-rules>
```

For example:

```
dn: cn=bob,o=engineering
```

becomes

```
dn: cn=bob,o=neworg
```

Example Command: If the placement rules are saved to a `pr1.xml` file, the following command instructs the utility to use the rules while processing the `1entry.ldf` file and to send the results to a destination file, `foutt1.ldf`.

```
ice -o -pfile://pr1.xml -SLDIF -f1entry.ldf -c -DLDIF -foutt1.ldf
```

1.11 Improving the Speed of LDIF Imports

In cases where you have thousands or even millions of records in a single LDIF file you are importing, consider the following:

- "Import Directly to a Server with a Read/Write Replica" on page 40
- "Use LBURP" on page 40
- "Configure the Database Cache" on page 40
- "Use Simple Passwords" on page 40
- "Use Indexes Appropriately" on page 40

1.11.1 Import Directly to a Server with a Read/Write Replica

If it's possible to do so, select a destination server for your LDIF import that has read/write replicas containing all the entries represented in the LDIF file. This will maximize network efficiency.

Avoid having the destination server chain to other eDirectory servers for updates. This can severely reduce performance. However, if some of the entries to be updated are only on eDirectory servers that are not running LDAP, you might have to allow chaining to import the LDIF file.

1.11.2 Use LBURP

The Novell Import Convert Export Wizard maximizes network and the eDirectory server processing efficiency by using LBURP to transfer data between the wizard and the server. Using LBURP during an LDIF import greatly improves the speed of your LDIF import.

For more information on LBURP, see Section 1.8, "LDAP Bulk Update/Replication Protocol," on page 30.

1.11.3 Configure the Database Cache

The amount of database cache available for use by eDirectory has a direct bearing on the speed of LDIF imports, especially as the total number of entries on the server increases. When doing an LDIF import, you might want to allocate the maximum memory possible to eDirectory during the import. After the import is completed and the server is handling an average load, you can restore your previous memory settings. This is particularly important if the import is the only activity taking place on the eDirectory server.

1.11.4 Use Simple Passwords

eDirectory uses public and private key pairs for authentication. Generating these keys is a very CPU-intensive process. With NDS eDirectory 8.5, you can choose to store passwords using the simple password feature of the Novell Modular Authentication Service (NMAS). When you do this, passwords are kept in a secure location in the directory, but key pairs are not generated until they are actually needed for authentication between servers. This greatly improves the speed with which an object that has password information can be loaded.

To enable simple passwords during an LDIF import, use the -l option with the destination handler.

If you choose to store passwords using simple passwords, you must use an NMAS-aware Novell® Client™ to log in to the eDirectory tree and access traditional file and print services. LDAP applications binding with name and password will work seamlessly with the simple password feature.

1.11.5 Use Indexes Appropriately

Having unnecessary indexes can slow down your LDIF import because each defined index requires additional processing for each entry having attribute values stored in that index. You should make sure that you don't have unnecessary indexes before you do an LDIF import, and you might want to consider creating some of your indexes after you have finished loading the data and after you have reviewed predicate statistics to see where they are really needed.

To ensure that your LDIF import is not slowed by indexing, you can turn your indexes off while bulk loading data. Once the data is loaded, you can turn your indexes on again.

For additional information , see “Indexes” in the eDirectory Technical Reference manual.

Other LDAP Utilities

2

The LDAP command line utilities can be called by your LDAP application to delete entries, modify entries, add entries, extend the schema, modify relative distinguished names, move entries to new containers, create search indexes, or perform searches. They all accept a -f <file> option which allows the processing of an LDIF file to perform a sequence of operations. The LDAP Libraries for C include the following utilities:

- Section 2.1, "ldapadd," on page 43
- Section 2.2, "ldapdelete," on page 45
- Section 2.3, "ldapmodify," on page 46
- Section 2.4, "ldapmodrdn," on page 48
- Section 2.5, "ldapsearch," on page 50
- Section 2.6, "ndsindex," on page 52

2.1 ldapadd

The ldapadd utility adds new entries. It has the following syntax:

```
ldapadd [options]
```

NOTE: On a NetWare server, the utility is called ladd.

If the -f option is specified, ldapadd reads the modifications from a file. If the -f option is not specified, ldapadd reads the modifications from stdin.

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapadd [options] > out.txt.

Replace [options] with one or more of the following:

Option	Description
-a	Adds new entries. The default for ldapadd is to add existing entries. This option can be used with the -f <file> option to read the entries to add from a file. A changetype should not be specified in the file.
-r	Replaces existing values. If invoked to replace entries, this flag must be set.
-c	Enables continuous operation mode. Errors are reported, but ldapadd will continue. The default is to exit after reporting an error
-f <file>	Reads the add information from an LDIF file instead of from standard input. The maximum length of a record is4096 lines. The record is assumed to be an LDIF record if the first line starts with a pound sign or a colon appears to the left of an equals sign.

Option	Description
-F	Forces the application to apply all changes regardless of the contents of input lines that begin with the following: `replica:` By default, these lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied.
Common Options	**Description**
-C	Enable referral following. (anonymous bind.)
-d <level>	Sets the LDAP debugging level to the specified level. The ldapmodify utility must be compiled with LDAP_DEBUG defined for this option to have any effect.
-D <binddn>	Specifies the dn to use in binding to the LDAP server. The dn should be a string-represented dn as defined in RFC 1779.
-e <file>	Specifies the certificate file to use with an SSL bind.
-h <host>	Specifies an alternate host on which the LDAP server is running.
-l <limit>	Specifies the connection timeout (in seconds).
-M	enable Manage DSA IT control. (non-critical)
-MM	enable Manage DSA IT control. (critical)
-n	Shows what would be done, but doesn't actually modify entries. Useful for debugging in conjunction with -v (verbose mode) which writes many diagnostics to standard output.
-p <port>	Specifies an alternate TCP port where the LDAP server is listening.
-P <version>	Specifies the LDAP version (2 or 3).
-v	Uses the verbose mode which writes many diagnostics to standard output.
-w <passwd>	Specifies the password to use for simple authentication.
-W	Prompts the user for the password for simple authentication. This is used instead of specifying the password on the command line.
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.

Option	Description
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

2.2 ldapdelete

The ldapdelete utility deletes the specified entry. It opens a connection to an LDAP server, binds, and then deletes. It has the following syntax:

```
ldapdelete [options] dn ...
```

NOTE: On a NetWare server, the utility is called ldelete.

The dn parameter is a list of distinguished names of the entries to be deleted. It interacts with the -f option in the following ways:

- If the -f option is missing from the command line and dn's are specified on the command line, the utility deletes the specified entries.
- If both dn and the -f option are in the command line, the utility reads the file for the dn's to delete and ignores any dn's in the command line.
- If both dn and the -f option are missing in the command line, the utility reads the dn from stdin.

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapdelete [options] > out.txt.

Replace [options] with one of the following:

Option	Description
-c	Enables continuous operation mode. Errors are reported, but ldapdelete will continue with deletions. The default is to exit after reporting an error.
-f <file>	Reads a series of dn's from the specified file. Each dn should be on its own line.
-r	Delete recursively
Common Options	**Description**
-C	Enable referral following. (anonymous bind.)
-d <level>	Sets the LDAP debugging level to the specified level. The ldapdelete utility must be compiled with LDAP_DEBUG defined for this option to have any effect.
-D <binddn>	Specifies the dn to use in binding to the LDAP server. The dn should be a string-represented dn as defined in RFC 1779.

Option	Description
-e <file>	Specifies the certificate file to use with an SSL bind.
-h <host >	Specifies an alternate host on which the LDAP server is running
-l <limit>	Specifies the connection timeout (in seconds).
-M	enable Manage DSA IT control. (non-critical)
-MM	enable Manage DSA IT control. (critical)
-n	Shows what would be done, but doesn't actually delete entries. Useful for debugging in conjunction with -v (verbose mode).
-p <port>	Specifies an alternate TCP port where the LDAP server is listening.
-P <version>	Specifies the LDAP protocol version: 2 or 3.
-v	Turns on verbose mode which writes diagnostics to standard output.
-w <passwd>	Specifies the password to use for simple authentication.
-W	Prompts for simple authentication. This is used instead of specifying the password on the command line.
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

2.3 ldapmodify

The ldapmodify utility modifies the attributes of an existing entry or adds new entries. It has the following syntax:

```
ldapmodify [options]
```

NOTE: On a NetWare server, the utility is called lmodify.

If the -f option is specified, ldapmodify reads the modifications from a file. If the -f option is not specified, ldapmodify reads the modifications from stdin.

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapmodify [options] > out.txt.

Replace [options] with one or more of the following:

Option	Description
-a	Adds new entries. The default for ldapmodify is to modify existing entries. If invoked to add entries, this flag must be set. This option can be used with the -f <file> option to read the entries to add from a file. A changetype should not be specified in the file.
-c	Enables continuous operation mode. Errors are reported, but ldapmodify will continue with modifications. The default is to exit after reporting an error
-r	Replaces existing values by default.
-f <file>	Reads the entry modification information from a file instead of from standard input. The maximum length of a record is 4096 lines. The record is assumed to be an LDIF record if the first line starts with a pound sign or a colon appears to the left of an equals sign. Each record should include a changetype line specifying add, modify, replace, or delete. (See "LDIF Examples" on page 55.)
-F	Forces the application to apply all changes regardless of the contents of input lines that begin with the following: `replica:` By default, these lines are compared against the LDAP server host and port in use to decide if a replog record should actually be applied.
Common Options	**Description**
-C	Enable referral following. (anonymous bind.)
-d <level>	Sets the LDAP debugging level to the specified level. The ldapmodify utility must be compiled with LDAP_DEBUG defined for this option to have any effect.
-D <binddn>	Specifies the dn to use in binding to the LDAP server. The dn should be a string-represented dn as defined in RFC 1779.
-e <file>	Specifies the certificate file to use with an SSL bind.
-h <host>	Specifies an alternate host on which the LDAP server is running.
-l <limit>	Specifies the connection timeout (in seconds).
-M	enable Manage DSA IT control. (non-critical)
-MM	enable Manage DSA IT control. (critical)
-n	Shows what would be done, but doesn't actually modify entries. Useful for debugging in conjunction with -v (verbose mode) which writes many diagnostics to standard output.

Option	Description
-p <port>	Specifies an alternate TCP port where the LDAP server is listening.
-P <version>	Specifies the LDAP version (2 or 3).
-v	Uses the verbose mode which writes many diagnostics to standard output.
-w <passwd>	Specifies the password to use for simple authentication.
-W	Prompts the user for the password for simple authentication. This is used instead of specifying the password on the command line
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

2.4 ldapmodrdn

The ldapmodrdn modifies the relative distinguished name of an entry. It can also move the entry to a new container. It has the following syntax

```
ldapmodrdn [options] [-s newSuperior] [dn newrdn]
```

Dn is a full LDAP dn. Newrdn is a new relative distinguished name.

If given, newrdn will replace the RDN of the entry specified by DN. If not given, the list of modification is read from stdin or from the file specified by "-f file".

NOTE: On a NetWare server, the utility is called lmodrdn dn <newrdn>).

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapmodrdn [options] > out.txt.

The following table describes these options and parameters.

Option	Description
-c	Enables continuous operation mode. Errors are reported, but ldapmodrdn will continue with modifications. The default is to exit after reporting an error.
-f <file>	Performs the sequences of modifications listed in the specified file. Each record consists of the distinguished name followed by the new rdn attribute on the following line. For example: dn1 rdn1 dn2 rdn2 Blank lines are ignored. This option cannot be used with the dn <newrdn> option.
-r	Specifies to remove the old rdn value from the entry. If not specified, defaults to keeping the old value.
-s <newSuperior>	Specifies the distinguished name of the container to which the entry is moving.
Common Options	**Description**
-C	Enable referral following. (anonymous bind)
-d <level>	Sets the LDAP debugging level to the specified level.
-D <binddn>	Specifies the dn of the entry to use in a bind operation.
-e <file>	Specifies the certificate file to use with an SSL bind.
-h <host>	Specifies an alternate host on which the LDAP server is running.
-l <limit>	Specifies the connection timeout (in seconds).
-M	enable Manage DSA IT control. (non-critical)
-MM	enable Manage DSA IT control. (critical)
-n	Shows what would be done, but doesn't actually rename the entry.
-p <port>	Specifies an alternate TCP on which the LDAP server is listening.
-P <version>	Specifies the LDAP version (2 or 3).
-v	Turns on verbose mode (sends diagnostics to standard output).
-w <passwd>	Specifies the password to use for simple authentication.
-W	Prompts the user for the password for bind operations.

Option	Description
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

2.5 ldapsearch

The ldapsearch utility searches the directory for specified attributes and object classes. It has the following syntax:

```
ldapsearch [options] filter [attribute list]
```

NOTE: On a NetWare server, the utility is called lsearch.

By default, ldapsearch will NOT follow referrals. To follow referrals, pass -C as an option. The ldapsearch utility now outputs in LDIF format, and binary attributes are printed in base64.

The filter must be an RFC-2254 compliant LDAP search filter. For more information, see “LDAP Search Filters” in *LDAP and eDirectory*.

The attribute list is a whitespace-separated list of the attributes to retrieve. The following values may replace the list:

- 1.1— no attributes
- *— all user attributes
- +— all operational attributes
- an empty list gets all non-operational attributes

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapsearch [options] filter [attribute list] > out.txt.

Replace options with one or more of the following:

Option	Description
-a <deref>	Specifies how to handle the dereferencing of an alias. It uses the following values: • Never—aliases are never dereferenced when locating the base object or searching. • Always—aliases are always dereferenced when locating the base object and searching. • Search—aliases are dereferenced when searching subordinates of the base object but not when locating the base object. • Find—aliases are dereferenced when locating the base object but not when search the subordinates of the base object.
-A	Retrieves attribute names only (no values).
-b <basedn>	Specifies the dn of the base container where the search should begin
-L	Prints entries in LDIF format.
-LL	Prints entries in LDIF format without comments.
-LLL	Prints entries in LDIF format without comments and version.
-s <scope>	Specifies the extent of the scope and use one of the following scope values: • base—search base object only • one—search the immediate subordinates of the base object • sub—search the base object and all of its subordinates
-S <attr>	Sorts the results by the specified attribute.
-t	Writes binary values to files in TMPDIR.
-tt	Writes all values to files in TMPDIR.
-T <path>	Write files to directory specified by path (default: "/tmp").
-u	Includes User Friendly entry names in the output.
-V	URL prefix for files.
-V <prefix>	Specifies the URL prefix for files (default: "file://tmp/").
-z <limit>	Specifies the maximum number of entries to return in the search.
Common Options	**Description**
-C	Enable referral following. (anonymous bind.)
-CC	Enable referral following. (authenticated bind with same bind DN and password.)
-d <level>	Sets LDAP debugging to the specified level.
-D <binddn>	Specifies the dn of the entry to use in bind operations.
-e <file>	Specifies the certificate file to use with an SSL bind.
-f <file>	Performs the sequence of searches listed in the specified file. Each line specifies the parameters for one search.

Option	Description
-h <host>	Specifies an alternate host on which the LDAP server is running.
-l <limit>	Specifies the connection and search timeout (in seconds).
-M	enable Manage DSA IT control. (non-critical)
-MM	enable Manage DSA IT control. (critical)
-n	Shows what would be done, but doesn't actually search the directory.
-p <port>	Specifies an alternate TCP port on which the LDAP server is listening.
-P <version>	Specifies the LDAP version (2 or 3).
-v	Turns on verbose mode (diagnostics to standard output).
-w <passwd>	Specifies the password to use for simple authentication
-W	Prompts the user for bind password.
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

2.6 ndsindex

The ndsindex utility creates, lists, suspends, resumes, or deletes indexes. It has the following syntax:

```
ndsindex [command] [options] [index]
```

For additional information on eDirectory indexes see “Indexes” in the *eDirectory Technical Overview*.

NOTE: On a NetWare server, the utility is called nindex.

TIP: Output from the ldap utilities is sent to stdout. If the utility exits before you can view the output, redirect the output to a file, for example, ldapadd [options] > out.txt.

[Command]

Replace [command] with one of the following index operations:

Command	Description
list	Lists the specified index(es).
add	Adds the specified index(es).
delete	Deletes the specified index(es).
suspend	Suspends the specified index(es) (Suspends the index to an off-line state).
resume	Resumes the specified index(es) (resumes the index from an off-line state).

[Options]

Replace [options] with one or more of the following options:

Options	Description
-D <binddn>	Specifies the dn to use in binding to the LDAP server. The dn should be a string-represented dn as defined in RFC 1779.
-e <file>	Specifies the certificate file to use with an SSL bind.
-h <host>	Specifies an alternate host on which the LDAP server is running.
-l <limit>	Specifies the connection timeout (in seconds).
-s	-Specifies the eDirectory Server DN.
-p <port>	Specifies an alternate TCP port where the LDAP server is listening.
-w <passwd>	Specifies the password to use for simple authentication.
-W	Prompts the user for the password for simple authentication. This is used instead of specifying the password on the command line
-Z	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the error is ignored and the operation continues. It is recommended that the -ZZ option be used in place of this option to cause the operation to abort if an error occurs. If a port is specified with this with this option, it must accept clear text connections. To verify the server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server will be accepted.

Options	Description
-ZZ	Starts TLS before binding to perform the operation. If an error occurs during the Start TLS operation the operation is aborted. If a port is specified with this with this option, it must accept clear text connections. To verify server identity, this option should be used in conjunction with the -e option to specify a server certificate file to validate the server trusted root certificate when TLS is started. If the -e option is not specified, any certificate from the server is accepted.

[Index]

For all index operations except add, replace [index] with the name of one or more indexes separated by a space.

For add index operations, replace [index] with one or more index definitions separated by a space, in the following format:

```
indexname;atrributename;matchingrule
```

Index Definition Element	Description
indexname	Specifies the name for the index.
attributename	Specifies the attribute to be indexed.
matchingrule	Specifies the index matching rule. Can be one of the following values: • PRESENCE: Presence matching optimizes queries that involve checking only for the presence of the specified attribute. • VALUE: Value matching optimizes queries that involve matching the entire attribute value or the first part of the attribute value. • SUBSTRING: Substring matching optimizes queries that involve matching a few characters or a substring of the attribute value. These indexes are the most costly to maintain, but provide the most flexibility.

To create a presence matching index for the cn attribute, [index] is replaced with the following:

Examples

```
ndsindex add [options] common_name;cn;PRESENCE
```

To suspend this index, specify the following:

```
ndsindex suspend [options] common_name
```

To resume this index, specify the following:

```
ndsindex resume [options] common_name
```

To delete this index, specify the following:

```
ndsindex delete [options] common_name
```

LDIF Examples

3

RFC 2849 "The LDAP Data Interchange Format (LDIF)—Technical Specification" describes the LDIF file format. This section contains information on basic LDIF formatting conventions and provides examples on how to perform the following tasks using LDIF:

3.1 Modifying the Attribute Values of an Entry

You use an LDIF change record to modify attribute values. A modify change record uses one or more of the following keywords.

Keyword	Description
add	Indicates that the values that follow should be added to the attribute.
delete	Indicates that the values that follow should be deleted from the attribute. Beware of the following conditions: • If no values follow, all values of the attribute are deleted. • If the attribute has no values, the operation fails.
replace	Indicates that the values that follow should replace the existing attribute values. Beware of the following conditions: • If no values follow, all values of the attribute are deleted. • If the attribute has no values, the operation succeeds. • If the attribute has any values, all values are deleted and replaced with the specified values.

Adding a Value. The following example adds a value of aperson@anorg.dom to the mail attribute.

```
version: 1

dn: cn=aperson,o=anorg,c=any
changetype: modify
add: mail
mail: aperson@anorg.dom
```

If the attribute is multi-valued, it adds the value. If the specified value already exists in the attribute, the operation fails.

Deleting a Value. The following example deletes the aperson@anorg.dom value from the mail attribute.

```
version: 1

dn: cn=aperson,o=anorg,c=any
changetype: modify
delete: mail
mail: aperson@anorg.dom
```

If the attribute does not contain the specified value, the operation fails. However, if the purpose was to ensure that the specified value did not exist in the attribute, the operation succeeds in achieving this result.

Deleting an Attribute. The following example deletes all values of the mail attribute.

```
version: 1

dn: cn=aperson,o=anorg,c=any
changetype: modify
delete: mail
```

If the attribute does not exist, the operation fails. However, if the purpose was to ensure that the attribute has no values, the operation succeeds in achieving this result.

Replacing a Value. The following example replaces the existing mail value with the specified value, aperson@anorg.dom.

```
version: 1

dn: cn=aperson,o=anorg,c=any
changetype: modify
replace: mail
mail: aperson@anorg.dom
```

If the attribute is multi-valued, all values are removed and the specified value is added. If the attribute doesn't exist, it is added with the specified value.

Making Multiple Modifications. You can specify multiple modifications in a single LDIF record. The following example adds a telephone number, deletes the fax telephone number attribute, and replaces the mail attribute.

A line containing only a hyphen (-) character is used to mark the end of an attribute entry.

```
version: 1

dn: cn=aperson,o=anorg,c=any
changetype: modify
add: telephonenumber
telephonenumber: +1 415 555 0002
-
delete: facsimileTelephoneNumber
-
```

```
replace: mail
mail: aperson@anorg.dom
-
```

The LDIF specification requires the ending hyphen (-) after the last attribute entry, but not all LDAP servers support it. The Novell LDAP server supports it.

3.2 LDIF Formatting Conventions

For complete details, see RFC 2849 “The LDAP Data Interchange Format (LDIF)—Technical Specification.” An LDIF file consists of a series of records. Each LDIF file begins with a Version record and then is followed either by a series of Change records or Content records. Change and Content records cannot be in the same file.

This section describes the following general LDIF formatting conventions:

- Version record
- Comment specifier
- Record separator
- Line folding
- Valid string characters

Version Record. Each LDIF file should start with a Version record. If no Version record is specified, any application processing the file is allowed to assume the file is version 0. Both the Novell Import Convert Export utility and the LDAP utilities support Version records. A Version record must be the first non-comment line in the file. It has the following format.

```
version: 1
```

Zero or more spaces are allowed after the colon.

Comment Specifier. Lines beginning with the pound (#) sign are treated as comments and are ignored when processing the file.

Record Separator. Records are separated by one or more blank lines (created by either a linefeed or carriage return/linefeed pair). The last record in the file must be terminated with a record separator (one or more blank lines).

Line Folding. Each record consists of a series of fields, one field per line. However, sometimes the information required by a field does not fit on one line. To fold a line, insert a line separator (either a linefeed or carriage return/linefeed pair) followed by a space. The LDIF parser processes this sequence by concatenating the lines and discarding the leading space.

Valid String Characters. Every operation takes a distinguished name. A colon following the specifier indicates that the name is in safe UTF-8 format, for example,

```
   dn: cn=aperson,o=anorg,c=any
```

Safe strings can begin with any character in the range 0x00 to 0x7F except for the following characters:

0x00 (NUL)
0x0A (LF)

0x0D (CR)
0x20 (space " ")
0x3A (colon ":")
0x3C ((less-than "<")

Subsequent characters in a safe string can be any character in the range 0x00 to 0x7F except the following:

0x00 (NUL)
0x0A (LF)
0x0D (CR)

Two colons following the specifier indicates that the name is a Base64 encoded name. The following example Base64 encodes the name Kim (which really doesn't require encoding because all of its characters are safe):

```
givenName:: a2lt
```

Any name or value that contains a prohibited character must be Base64 encoded. For more information on Base64 encoding, see Section 5.2 of RFC 1521.

3.3 Adding Entries

You use can LDIF change record to add entries. An add change record requires values for all of the mandatory attributes of the entry's base class. The objectClass attribute must be set to the base class and may also be set to the super classes of the base class.

Since eDirectory maintains referential integrity, an LDIF file with multiple add change records must first add the container entries and then the leaf entries that will reside in the container. Leaf entries cannot be added to containers that don't already exist. (See the forward referencing option in the Novell Import Convert Export utility if you have an existing file with invalid ordering.)

The following example adds two organizational units (a college and department) under the UofZ organization and then adds an inetOrgPerson.

```
version: 1

dn: ou=Humanities, o=UofZ
changetype: add
objectClass: top
objectClass: ndsLoginProperties
objectClass: organizationalUnit
telephoneNumber: +1 415 555 0000

dn: ou=English, ou=Humanities, o=UofZ
changetype: add
objectClass: top
objectClass: ndsLoginProperties
objectClass: organizationalUnit
telephoneNumber: +1 415 555 1111

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
changetype: add
sn: Jones
```

```
givenName: Kim
objectClass: top
objectClass: ndsLoginProperties
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
telephoneNumber: +1 415 555 2222
mail: Kim.Jones@UofZ.edu
userPassword: Kim1234
```

3.4 Deleting Entries

You can use an LDIF change to delete entries. A delete change record requires two items: the distinguished name of the entry to delete and the delete changetype. Container entries cannot be deleted unless they are empty.

The following example deletes the entries created in the adding entries example. The entries are deleted in reverse order of adding so that the container entries are empty.

```
version: 1

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
changetype: delete

dn: ou=English, ou=Humanities, o=UofZ
changetype: delete

dn: ou=Humanities, o=UofZ
changetype: delete
```

3.5 Renaming or Moving an Entry

You can use an LDIF change record to rename an entry, move it, or do both in one operation. Modifying the DN uses two keywords and moving an entry requires a third keyword. These keywords have the following meaning:

Keyword	Description
newrdn	Indicates that this line contains the new RDN for the entry.
deleteoldrdn	Indicates whether the old RDN value should be deleted: 0=save, 1=delete. When the old value is saved, it becomes an additional attribute value but is no longer part of the entry's distinguished name.
newsuperior	Indicates that this line contains the DN of the new container for the entry.

Renaming an Entry. The following example shows how to rename an entry and retain the old value.

```
version: 1

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
```

```
changetype: moddn
newrdn: cn= Kim Jones-Smith
deleteoldrdn: 0
```

Moving an Entry. The following example shows how to move an entry from the English department to Spanish department.

```
version: 1

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
changetype: moddn
newrdn: cn=Kim Jones
deleteoldrdn: 1
newsuperior: ou=Spanish, ou=Humanities, o=UofZ
```

Renaming and Moving an Entry. The following example shows how to rename and move an entry in the same operation.

```
version: 1

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
changetype: moddn
newrdn: cn=Kim Jones-Smith
deleteoldrdn: 0
newsuperior: ou=Spanish, ou=Humanities, o=UofZ
```

3.6 Adding eDirectory Indexes

Indexes are used to sort the entries in the replicas of the local server according to specified attributes. They significantly increase search performance and allow the display of sorted results. NDS 8 and above ship with predefined indexes for commonly used attributes. NDS eDirectory 8.5 allows you to specify additional attributes for indexing.

However, each additional index adds overhead to update operations. Before adding an index, you should check the Predicate Data on the server to see if any of the attributes your application is using would benefit from an index. For more information, see the *Administration Guide* for eDirectory eDirectory 8.7.3.

To specify an index in an LDIF file, you must supply values for following case ignore strings which are separated by a dollar ($) sign.

Order	String	Description
1	Index version	Reserved for future use. In NDS eDirectory 8.5, this should always be set to zero (0).
2	Index name	Specifies the user-defined name for the index, such as “Family Name” or “Zip Code.” The string should not contain the dollar ($) sign.

Order	String	Description
3	Index state	Specifies the state of the index. When defining an index, this field should be set to 2 (online). eDirectory supports the following values: • 0—suspended which indicates the index is not used in queries and is not updated. • 1—bringing online which indicates the index is in the process of being created. • 2—online which indicates the index is up and working. • 3—pending creation which indicates the index has been defined and is waiting for the background process to run. The background process changes the state once the building begins.
4	Index rule	Specifies the type of matching: • 0—Value matching which optimizes queries that involve the entire value or the first part of the value, for example, a query for all entries with a surname equal to Jensen or begin with Jen. • 1—Presence matching which optimizes queries that involve only the presence of an attribute, for example, a query for all entries with a surname attribute. • 2—Substring matching which optimizes queries that involve a match of a few characters, for example, a query for all entries with a surname containing “der”. This query returns entries with the surnames of Derington, Anderson, and Lauder.
5	Index type	Specifies who created the index. When defining an index, you must set this value to 0. eDirectory supports the following values: • 0—user defined • 1—added on attribute creation • 2—required for operation • 3—system index
6	Index value state	Specifies the source of the index. When defining an index, set this string to 1. eDirectory supports the following values: • 0—uninitialized • 1—added from server • 2—added from local DIB • 3—deleted from local DIB • 4—modified from local DIB
7	Attribute name	Specifies the NDS name for the attribute. Many attributes in eDirectory have both an LDAP name and an NDS name. This string requires the NDS name.

The following example shows how to add an index for the mail attribute (which has an NDS name of EMail Address).

```
version: 1

dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
```

```
changetype: modify
add: indexDefinition
indexDefinition: 0$myEmailIndex$2$1$0$1$Email Address
```

3.7 Modifying the Schema

You can use LDIF files to add attribute and class definitions. eDirectory also allows you to modify existing schema definitions.

NOTE: Modifying existing definitions should be done with caution since you are changing an attribute or class that another application may be using.

You can modify the following features of existing class definitions:

- Containment rules. You can add classes to or delete classes from the definition's containment rules.
- Super class rules. You can add a new super class to an object class, but you cannot remove any defined super classes.
- ASN.1 IDs. You can modify the ASN.1 ID assigned to the class.
- Class flags. You can change a leaf class definition into a container class by turning on the container class flag. But once turned on, it cannot be removed.
- Optional attributes. You can add new optional attributes to or delete optional attributes from the class definition. This is not the preferred method. If possible, create an auxiliary class and use the auxiliary class to add the attributes to the entries created from the class definition.

You can modify the following features of existing attributes:

- On sized attributes, you can increase the upper boundary.
- You can set the following attribute flags:

  X-NDS_NEVER_SYNC
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE
  X-NDS_PUBLIC_READ
  X-NDS_SERVER_READ
- You can clear the following attribute flags:

  SINGLE-VALUE
  X-NDS_LOWER_BOUND
  X-NDS_NEVER_SYNC
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE
  X-NDS_PUBLIC_READ
  X-NDS_SERVER_READ

Adding an Attribute Definition. To add an attribute definition, you modify the subschemaSubentry object which always has a DN of cn=schema in NDS. You obtain this name by reading the root DSE.

The following example adds five attributes that could be used for a student class definition. In the example, each structural element in attributeTypes uses line folding (new line plus one space) so that the various elements are more visible. (The OIDs are only sample ASN.1 IDs and have not been registered.)

```
version: 1

dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: (
  1.20.300.400.500.4.1
  NAME 'advisor'
  DESC 'faculity advisor for the student'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
  SINGLE-VALUE
  X-NDS_PUBLIC_READ '1'
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE '1')
attributeTypes: (
  1.20.300.400.500.4.2
  NAME 'major'
  DESC 'students major accepted by the department'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE '1')
attributeTypes: (
  1.20.300.400.500.4.3
  NAME 'minor'
  DESC 'students minor accepted by the department'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE '1')
attributeTypes: (
  1.20.300.400.500.4.4
  NAME 'department'
  DESC 'department supervising the student'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
  SINGLE-VALUE
  X-NDS_PUBLIC_READ '1'
  X-NDS_NOT_SCHED_SYNC_IMMEDIATE '1'
)
attributeTypes: (
  1.20.300.400.500.4.5
  NAME 'studentID'
  DESC 'unique ID assigned to the student'
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.36
  SINGLE-VALUE)
```

The values 1.20.300.400.500.4.1 through 1.20.300.400.500.4.5 specify the OID or ASN1 ID of the attribute. For information on obtaining an OID and a naming prefix that guarantees uniqueness in the eDirectory schema, see "Schema Extensions" in *eDirectory Schema Reference*.

The NAME element specifies the name of the attribute.

The DESC element describes the attribute.

The SYNTAX element specifies the type of data the attribute can contain. For a list of supported OIDs, see "Attribute Syntax Definitions" in *eDirectory Schema Reference*.

The SINGLE-VALUE flag specifies that the attribute is single valued. When not specified, the attribute defaults to multi-valued. For a description of the possible LDAP and X-NDS flags, see "Attribute Flags" in *LDAP and eDirectory*.

Adding a Structural Class Definition. To add a class definition, you modify the schema object, dn: cn=schema. You obtain this name by reading the root DSE.

NOTE: eDirectory maintains referential integrity, so any new attributes must be added in the file before they are added to the class. The following example uses attributes that need to be added to the schema before the class definition.

In the following example, each structural element in objectClasses uses line folding (new line plus one space) so that the various elements are more visible.

```
version: 1

dn: cn=schema
changetype: modify
objectClasses: (
  1.20.300.400.500.6.1
  NAME 'student'
  SUP 'person'
  STRUCTURAL
  MUST (cn $ studentID $ givenName $ sn)
  MAY (advisor $ major $ minor $ department)
  X-NDS_NAMING ('cn')
  X-NDS_CONTAINMENT ('organization' 'organizationalUnit'
  'domain' )
  X-NDS_NOT_CONTAINER '1'
  X-NDS_NONREMOVABLE '1')
```

The 1.20.300.400.500.6.1 value specifies the OID or ASN1 ID of the class. For information on obtaining an OID and a naming prefix that guarantees uniqueness in the eDirectory schema, see "Schema Extensions" in *eDirectory Schema Reference*.

The NAME element specifies the name of the object class.

The SUP element specifies the super class of the object class.

The STRUCTURAL element specifies that the class is an effective class. ABSTRACT specifies a non-effective class, and AUXILIARY specifies an auxiliary class.

The MUST element specifies the mandatory attributes.

The MAY element specifies the optional attributes.

For a description of the possible X-NDS flags, see "Object Class Flags" in *LDAP and eDirectory*.

NOTE: This example shows how to add a structural class. Since student is a class that might apply to more than one type of user in the directory (inetOrgPerson or User), this class might be more useful as an auxiliary class. See the next example.

Adding an Auxiliary Class. The following example creates two new attributes, creates an auxiliary class with these new attributes, and then adds an inetOrgPerson entry with the auxiliary class as an object class of the entry and with values for the auxiliary class attributes.

```
version: 1

# Add an attribute to track a bear's hair. The attribute is
# multi-valued, uses a case ignore string syntax,
# and has public read rights
# Values may include: long hair, short, curly, straight,
# none, black, and brown
# X-NDS_PUBLIC_READ ’1’ The 1 allows public read,
# 0 denies public read
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.10 NAME ’bearHair’ SYNTAX
1.3.6.1.4.1.1466.115.121.1.15 X-NDS_PUBLIC_READ ’1’ )

# add an attribute to store a bear's picture
dn: cn=schema
changetype: modify
add: attributeTypes
attributeTypes: ( 2.16.840.1.113719.1.186.4.11 NAME ’bearPicture’
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE )

# create an Auxiliary class for the bearfeatures
dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: (2.16.840.1.113719.1.186.6.101 NAME ’bearFeatures’ MAY
(bearHair $ bearPicture) AUXILIARY)

# now create a user named booboo
dn: cn=booboo,o=jellystone
changetype: add
cn: booboo
sn: bear
givenName: booboo
bearHair: Short
bearHair: Brown
bearHair: Curly
bearPicture:< file:///c:/tmp/alien.jpg
objectClass: top
objectClass: person
objectClass: inetOrgPerson
objectClass: bearFeatures
```

3.8 Using Controls

A control can be specified in an LDIF file using the following format:

```
control: <oid> <TRUE/FALSE>[::base64 encoded data]
```

oid	Oid identifying the control.
TRUE/FALSE	Criticality. If the server does not recognize the control, or it is not appropriate for the operation, TRUE will cause the server to fail the operation. FALSE will cause the server to ignore the operation.
::base64 encoded data	Optional. Any data that might be required by this control, usually encoded in base64 format.

For example, the following LDIF snippet includes the DSA control to enable operations with superior referrals:

```
dn:dc=Acme,dc=com
control: 2.16.840.1.113730.3.4.2 TRUE
changetype: add
...
```

3.9 Using Content Records

Content records are the same as change records except that they do not have a changetype specifiers. They are usually generated to list the current entries in a directory and the entry's attributes and values. Content records and change records cannot be placed in the same LDIF file.

Both the Novell Import Convert Export utility and the ldapmodify utility have options that allow content records to be used as add change records.

- The -a option of the LDIF source handler of the Novell Import Convert Export utility changes content records into change records with a changetype of add.
- The -a option of the ldapmodify utility changes content records into change records with a changetype of add.

The following examples of content records could be used by either utility to add these entries to the directory.

```
version: 1

dn: o=UofZ
objectClass: top
objectClass: ndsLoginProperties
objectClass: organization

dn: ou=Humanities, o=UofZ
objectClass: top
objectClass: ndsLoginProperties
objectClass: organizationalUnit
telephoneNumber: +1 415 555 0000

dn: ou=English, ou=Humanities, o=UofZ
objectClass: top
objectClass: ndsLoginProperties
objectClass: organizationalUnit
telephoneNumber: +1 415 555 1111
```

```
dn: cn=Kim Jones, ou=English, ou=Humanities, o=UofZ
sn: Jones
givenName: Kim
objectClass: top
objectClass: ndsLoginProperties
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
telephoneNumber: +1 415 555 2222
mail: Kim.Jones@UofZ.edu
userPassword: Kim1234
```

Revision History

A

The following table lists all changes made to the LDAP Tools documentation:

March 2006	Fixed formatting issues.
February 2004	Renamed the product name from “NDS” to “Novell eDirectory” at relevant instances
October 2003	Added the following: • -k and -g options for LDIF source and destination • -y option for LDAP source and destination • A new placement example
March 2003	Added information on the Section 2.6, “ndsindex,” on page 52 utility.
September 2002	Made the following changes: • Separated the LDAP tools information into a separate book • Added information on Section 3.8, “Using Controls,” on page 65 in LDIF files • Revised and enhanced the Section 1.6, “DirLoad Driver,” on page 20 section • Updated the ldapsearch utility to use LDIF output by default