CA Directory

Reference Guide

r12.0 SP6

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Provide Feedback

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

If you would like to provide feedback about CA Technologies product documentation, complete our short customer survey, which is available on the CA Support website at http://ca.com/docs.

Character Limitations and Special Characters

CA Directory requires that the names of computers, users, directories, and so on, are valid for the operating system and also adhere to the following restrictions:

- **Installation Location for CA Directory ($DXHOME or %DXHOME%)**—The name of the location (folder or directory) that CA Directory is installed into must contain only alphabetic and numeric characters.
- **DXlink Password in the Knowledge File (ldap-dsa-password)**—If the password that you specify for *ldap-dsa-password* in the knowledge file contains a backslash (\), you must escape it with a second backslash (\\).

  For example, if the actual password is *p3.M\b@)*, you must include it in the knowledge file like this:

  ```
  ldap-dsa-password = "p3.M\\b@)"
  ```

- **Other Character Limitations**—CA Directory requires that the following items include only the standard ASCII characters that appear in English:
  - DSA names
  - Directory prefixes, for example, <c AU><o DEMOCORP>

Command Formatting Conventions

In this guide, commands are shown in a different font from the main text, as in this example:

```
get dynamic-group;
```

Variables that you must replace appear in italic text. In this example, replace *assoc-number* with the actual association number:

```
abort user assoc-number;
```

If you must enter only one of a list of options, the options are shown separated by the pipe character |. In this example, you should choose *either* true *or* false:

```
set access-controls = true | false;
```

Optional items are shown enclosed in square brackets, as the *tag* option is in this example:

```
set admin-user [tag] = own-entry
```

If items can be repeated, this is shown by a trailing ellipsis ..., for example:

```
item 1 [,item 2 ...]
```

File Location Convention

This document refers to the CA Directory installation location as DXHOME. For example, the location DXHOME/config/schema represents the following locations in a default installation:

- **Windows—**C:\Program Files\CA\Directory\dxserver**\config\schema**
- **UNIX—**/opt/CA/Directory/dxserver**/config/schema**

Format of Distinguished Names

The X.500 and LDAP communities differ in the way they write distinguished names (DNs):

- **X.500**—DNs are written from the top of the tree down, for example:

  ```
  <c US><o Acme><ou Staff><cn "John Citizen">
  ```

- **LDAP**—DNs are written from the leaf entry up, for example:

  ```
  cn=John Citizen, ou=Staff, o=Acme, c=US
  ```

If a portion of prefix is more than one word, you can enclose the whole prefix in quotes or just the problem portion. For example, both of these prefixes will work:

```
o=“democorp test”,c=au
“o=democorp test,c=au”
```

Note: Use a pair of quotes (“”) for a null DN.

Contents

Chapter 11: The Directory User Agent (DUA) 283

Chapter 1: Internal Commands for DSA, DUA, and Scripting

Internal commands are executed by the Command Line Interpreter itself.

This section contains the following topics:

Syntax for Commands

The following command executes a script file:

```
source "script-file";
```

The *source* command is relative to the file that invokes it. For example, you can source the file *schema.dxg* using the following command:

```
source "../schema/schema.dxg";
```

The schema.dxg script can then source a file in its own directory using the following command:

```
source "attr.dxc";
```

To include a comment, begin the comment line with the # symbol.

Script File Errors and Debugging

Usually the system does not echo commands in script files before they are executed. If there is an error in the script file, a message similar to the following is displayed:

```
>>>> set allow-binds = 1;
 ----------------------^ (line 12 in 'test.dxc')
 Syntax Error: Expected 'true' or 'false'
```

To obtain a full log of every command executed in a script file, set the first lines of the script file to:

```
echo-on;
set trace-file = "script.log";
```

After running the script file, you can examine the log file to locate any failed command and the reason for the failure.

! Command

The ! command is the short form of the *source* command.

For example, the following two lines are equivalent:

```
source "democorp.dxc";

! "democorp.dxc";
```

echo Command

The *echo* command displays each command in a script file before the command is executed.

The output of echoed commands goes to the DSA console (if connected) and the trace-log file (if open).

Messages can be displayed using the echo command, for example:

```
echo "Initializing ...";
```

echo-off Command

The *echo-off* command turns off echoing. After this command has been read, no output is sent to the log file.

If you do not want the password to be displayed when it is entered, use the *echo-off* command in the Telnet session.

echo-on Command

The *echo-on* command sends an echo of the command to the command before it is executed, and also send the command results to the same output.

To obtain a full log of a script on a DSA, set the first lines of the script file to:

```
echo-on;
set trace-log = "script.log";
```

script.log is the name of the file to which the output is sent.

To obtain a full log of a script on a DUA, do the following:

1. Set the first lines of the script file to:

   ```
   echo-on;
   set log-type = "script.log";
   ```

2. Set the last lines of the script to the following:

   ```
   flush trace-log;
   quit;
   ```

source Command

The *source* command includes the contents of another file.

You can nest source commands, and source a file that contains further source commands.

The source command is relative to the file that invokes it.

```
source filename;
```

filename

Defines another configuration file.

Example

If a file contains the first command listed below, it sources the file *schema.dxg*. The *schema.dxg* script can then source a file in its own directory using the second command:

```
source "../schema/schema.dxg";

source "attr.dxc";
```

Chapter 2: DSA Commands (DXserver Commands)

The DXserver commands control the way the DSAs work.

You can use these commands in these ways:

- Use the commands in *configuration files*.

  The command takes effect when the DSA is restarted or reinitialized.

- Enter the commands through the DSA console

  The command takes effect immediately. When the DSA is restarted or reinitialized, the command will no longer take effect.

- Include the commands in *script files*.

  The command takes effect when the script file is run. When the DSA is restarted or reinitialized, the command will no longer take effect.

This section contains the following topics:

About the CA Directory Commands

The commands, flags, and settings control every aspect of CA Directory configuration and management.

Quotation Marks

You do not need to enclose simple (one-word) strings in quotation marks.

However, you must quote complex (many-word) strings. Within a quoted string, you can use a backslash (\) as an escape mechanism for the following cases:

Reason for Using Quotes	Example	String
String already contains quotes	my “favorite” car	"my \"favorite\" car"
String already contains a backslash	c:\myfile	"c:\\myfile"
String contains UTF-8 characters	Jürgen Weiß	"J\xC3\xBCrgen Wei\xC3\9Fx"

Non-ASCII Characters

You can specify attribute values in character sets other than ASCII. LDAPv3 uses UTF-8 for internationalization.

To use a UTF-8 character in an attribute value, do the following:

- Precede the value with *utf8*
- Escape each two-character encoding with *\x*
- Enclose the string in double quotes

Example: Search for All Entries with Common Names Beginning with é

In UTF-8, the character *é* (e-acute) is C3A9.

To search for all entries with common names beginning with *e*, use the following DUA command:

```
search-req base-object=<> whole-subtree filter = { attr = commonName substrings [
initial e };
```

To search for all entries with common names beginning with *é*, use the following DUA command:

```
search-req base-object=<> whole-subtree filter = { attr = commonName substrings [
initial utf8 "\xC3\xA9" };
```

The Latin-1 character e-acute (E9) in UTF-8 (C3A9) has been encoded.

More information:

Non-ASCII Characters in LDIF Files (see page 245)

Command History

DXserver stores commands when they are entered. You can view these commands by using the history command, and can repeat them by using the number of the command in the history list.

Commands Categorized by Function

This section lists all of the DXserver commands, categorized by function.

Each of these commands is described fully in Commands Reference (see page 27).

Commands for Access Controls

Use the following commands to enable and configure access controls:

- set admin-user Command (see page 48)
- set group Command (see page 78)
- set protected-items Command (see page 111)
- set public-user Command (see page 116)
- set reg-user Command (see page 122)
- set super-user Command (see page 130)

Commands for Using DSA Memory

Use the following commands to enable and configure caching:

- set cache-index Command (see page 62)
- set cache-reverse Command (see page 62)
- set lookup-cache Command (see page 82)

Commands for Logging

Use the following commands to enable and configure logging:

- close log Command (see page 31)
- flush log Command (see page 35)
- get log Command (see page 40)
- set force-flush-all Command (see page 77)
- set log Command (see page 81)
- set time-log-search-threshold Command (see page 132)
- set time-log-update-threshold Command (see page 133)
- set update-log-show-values Command (see page 138)

Commands for Managing User Accounts

Use the following commands to enable and configure password management:

- set password-age Command (see page 92)
- set password-age-warning-period Command (see page 93)
- set password-allow-ignore-expired Command (see page 93)
- set password-allow-ignore-suspended Command (see page 94)
- set password-allow-locking Command (see page 94)
- set password-alpha Command (see page 95)
- set password-alpha-num Command (see page 95)
- set password-enforce-quality-on-reset Command (see page 96)
- set password-force-change Command (see page 97)
- set password-grace-logins Command (see page 98)
- set password-history Command (see page 98)
- set password-last-use Command (see page 99)
- set password-lowercase Command (see page 99)
- set password-max-length Command (see page 100)
- set password-max-repetition Command (see page 100)
- set password-max-substring-repetition Command (see page 100)
- set password-max-suspension Command (see page 101)
- set password-mimic-netscape-response-controls Command (see page 102)
- set password-min-age Command (see page 103)
- set password-min-length Command (see page 103)
- set password-min-length-repeated-substring Command (see page 104)
- set password-netscape-op-attrs Command (see page 104)
- set password-non-alpha Command (see page 105)
- set password-non-alpha-num Command (see page 105)
- set password-numeric Command (see page 106)
- set password-policy Command (see page 106)
- set password-proxy-user Command (see page 107)
- set password-retries Command (see page 107)
- set password-storage Command
- set password-substring-attrs Command (see page 108)

- set password-suspended-trap Command (see page 109)
- set password-uppercase Command (see page 109)
- set password-username-substring Command (see page 110)

Commands for Schemas

Use the following commands to enable and configure password management:

- clear schema command (see page 30)
- get schema command (see page 42)
- set attr-set command (see page 59)
- set attribute command (see page 60)
- set name-binding command (see page 88)
- set object-class command (see page 90)
- set oid-prefix command (see page 91)
- set syntax-alias command (see page 132)
- set unique-attrs command (see page 137)

Commands Reference

abandon Command—Cancel Operations

The *abandon* command cancels any outstanding operations on one or more bindings.

To use this command, you need to know the binding number, which you can find out using the *get users* command. You might also need to know the operation number, which is the invoke-id seen in the trace logs.

This command has the following format:

```
abandon { all | operation operation-number } on association binding-number;
```

operation-number

Specifies the operation that you want to cancel.

assoc-number

Specifies the binding which contains the operation you want to cancel.

Example: Abandon Unfinished Operations

The following command abandons all unfinished operations on the binding 131072:

```
abandon all on association 131072;
```

The following command abandons only operation 2 on binding 131072:

```
abandon operation 2 on association 131072;
```

More information:

get users Command—Display the Current Bindings (see page 43)

Outcome of the abandon Command

When an operation is abandoned, the following messages are sent:

- The targeted service returns the following message:

  ```
  Service error: operation abandoned
  ```

- The abandoned operation returns the following message:

  ```
  abandon confirm
  ```

If the operation cannot be abandoned, the operation returns the following message, with the appropriate error message explaining why the operation was not abandoned:

```
abandon-refused
```

abort Command—Close Bindings

The *abort* command closes all or one binding. The DSA discards any outstanding operations when it aborts a binding.

This command has the following format:

```
abort all-users | user binding-number;
```

all-users

Aborts all bindings.

user *binding-number*

Aborts one binding. *binding-number* is a number that specifies the binding you want to abort. Use the *get users* command to find this.

Example: Abort All Operations

The following command aborts all operations on the binding 131072:

```
abort user 131072;
```

clear access Command—Remove All Static Access Controls

The *clear access* command removes all of the static access controls.

This command is often used before loading access controls, to ensure that conflicting controls are not loaded.

This command has the following format:

```
clear access;
```

More information:

get access Command—Display Access Control Settings (see page 36)

clear allow-search Command—Remove Search Profile Definitions

The *clear allow-search* command removes all search profile definitions from the DSA.

Before you define a search profile, you should delete any existing search profile that has the same name. To do so, you need to delete all search profiles.

This command has the following format:

```
clear allow-search;
```

clear dsas Command—Remove Knowledge of all Remote DSA Definitions

The *clear dsas* command removes knowledge of all remote DSA definitions.

This command is often used before sourcing knowledge of remote DSAs, to prevent conflicting definitions of remote DSAs being loaded.

Note: This command is only useful for DSAs that use the pre-r12 configuration.

For information about remote DSA connections, see Remote Connections.

This command has the following format:

```
clear dsas;
```

clear dynamic-group Command

This command has the following format:

```
clear dynamic-group;
```

clear multi-write-queue Command—Remove Entries from Multi-Write Queue

The *clear multi-write-queue* command removes all entries from the multi-write queue of the DSA you have indicated.

This command has the following syntax:

```
clear multi-write-queue DSA_Name;
```

clear referential-integrity Command—Remove Referential Integrity Rules

Use the *clear referential-integrity* command to remove all the referential integrity rules.

This command has the following format:

```
clear referential-integrity;
```

clear schema Command—Remove All Schema Information

The *clear schema* command removes all of the schema information.

This command is often used before loading the schema, to ensure there are no conflicting definitions for attributes or object classes.

For information about remote DSA connections, see Remote Connections.

This command has the following format:

```
clear schema;
```

clear view Command—Remove All View Definitions

The *clear view* command removes all view definitions from the DSA.

Before you define a view, you should delete any existing view that has the same name. To do so, you need to delete all views.

This command has the following format:

```
clear view;
```

close log Command—Stop Output Being Sent to a Log File

The *close* command stops output to the log file by closing it.

When a DSA shuts down, it automatically closes any open log file. To close a log file manually, use the *close* command. You can only close one log at a time.

Note: The alarm log cannot be closed.

This command has the following format:

```
close log-type;
```

log-type

alert-log

cert-log

connect-log

diag-log

query-log

snmp-log

stats-log

summary-log

trace-log

update-log

warn-log

Example: Close The Trace Log

```
close trace-log
```

dump dxgrid-db Command—Take a Consistent Snapshot Copy of a Datastore

The *dump dxgrid-db* command takes a consistent snapshot copy of the datastore of a running DSA (an online dump). The DSA completes any updates before carrying out this command and does not start any more updates until the copy is finished.

The datastore file is copied to a file with an extension starting .z so the database file is *dxgrid-db*.zdb.

Note: Each dump overwrites the previous backup file. Create a cron job on UNIX or a scheduled task on Windows to copy the backed up file to a safe location before the next dump.

The DXdumpdb tool can export data from a datastore created by the dump command.

The command has the following format:

```
dump dxgrid-db [period start period];
```

period *start period*

(Optional) Specifies that the online dump is performed at regular intervals.

start

Defines the number of seconds from Sunday 00:00:00 a.m. GMT.

Note: The start time is defined using GMT and not your local time.

period

Defines the number of seconds between online dumps.

Note: The start time is relative to the period and should be lower than it. If you specify a start time that is greater than the period, the actual start is *start* - *period*. For example, if the *period* is 3600 seconds (one hour) and *start* is 3610 seconds, the online dump starts 10 seconds from midnight GMT and continues every hour from then on.

Example: Perform an Online Dump Every Hour

The following command takes a snapshot copy of the datastore every hour:

```
dump dxgrid-db period 0 3600
```

Example: Perform an Online Dump Every Night

The following command takes a snapshot copy of the datastore every night at 3 a.m. in a GMT+10:00 time zone:

```
dump dxgrid-db period 61200 86400
```

In this example, the start time is the number of seconds from Sunday midnight GMT to the first 3 a.m. slot, corrected by the time zone value, as follows:

```
(3 am – 10 time zone + 24 hours) * 60 minutes * 60 seconds = 61,200
```

The 24 hour period is calculated as follows:

```
24 hours * 60 minutes * 60 seconds = 86,400
```

dxserver Command—Start, Stop, Install, or Initialize DSAs

The *dxserver* command starts, stops, installs, initializes, or reports on one or all DSAs.

This command has the following format:

```
dxserver options;
```

forcestop *dsaname*

Stops the DSA *dsaname* even if it has multiwrite queues.

init *dsaname* | all

Signals one or all DSAs *dsaname* to reread the configuration files (if running). For example:

```
dxserver init DemoCorp-Host5-democorp
```

install *dsaname*

Adds *dsaname* to the list of servers to start at system startup. For example:

```
dxserver install DemoCorp-Host5-democorp
```

Note: On Windows, the *dxserver start* command implicitly does an install.

remove *dsaname*

Removes *dsaname* from the auto-startup list, for example:

```
dxserver remove DemoCorp-Host5-democorp
```

Note: You must stop the DSA before you remove it.

start *dsaname* | all

Starts one or all DSAs. For example:

```
dxserver start DemoCorp-Host5-democorp
```

status *dsaname*

Reports the running status of a DSA. For example:

```
dxserver status DemoCorp-Host5-democorp
```

If you omit *dsaname*, the status of all DSAs is reported.

stop *dsaname* | all

Stops one or all DSAs. For example:

```
dxserver stop DemoCorp-Host5-democorp
```

version

Displays information about the version of CA Directory that is installed.

flush log Command—Force All Output to Be Written to a Log File

The flush command forces all buffered output to a particular log file. This lets you examine a current log file off-line while the normal log file is still open.You can only flush one log at a time.

Note: Alarm logs are always flushed.

This command has the following format:

```
flush log-type;
```

log-type

alert-log

cert-log

connect-log

diag-log

query-log

snmp-log

stats-log

summary-log

time-log

trace-log

update-log

warn-log

Example: Flush The Trace Log

```
flush trace-log
```

force-shutdown Command—Force a DSA to Shut Down

The *force-shutdown* command stops a DSA that has a multiwrite queue. Use this command in the DSA console.

This command has the following format:

```
force-shutdown;
```

get access Command—Display Access Control Settings

The *get access* command displays the current access control settings.

This command has the following format:

```
get access;
```

get agreement Command—Display a DISP Agreement

The *get agreement* command displays the DISP configuration details for a particular agreement.

This command has the following format:

```
get agreement id.version | agreements;
```

id

Specifies the identification number of the agreement

version

Specifies the version number of the agreement

agreements

Displays all agreements

Example: Output from the *get agreement* Command

```
Agreement 1.0
    initiator   = EAGLE supplier
    responder   = BACKUP
    area= <countryName AU><organizationName Democorp>
    - last update at Thu Jul 9 09:53:47 2006
```

get agreements Command—Display All DISP Agreements

The *get agreements* command displays all DISP agreements configured for the DSA.

This command has the following format:

```
get agreements;
```

get allow-search Command—Display Search profiles

The *get allow-search* command displays a list of the DSA's search profiles and the current default search profile.

This command has the following format:

```
get allow-search;
```

More information:

set allow-search-default Command—Make One Search Profile the Default (see page 58)
set allow-search Command—Define a Search Profile (see page 56)

get cache Command—Display the Cache Configuration

The *get cache* command displays the cache configuration variables and their values.

It provides statistical information about the cache configuration.

This command has the following format:

```
get cache;
```

get ciphers Command—List All Supported Ciphers

The *get ciphers* command lists all the ciphers that are supported on the running DSA. These are the ciphers that can be used for SSL and TLS connections.

This command has the following format:

```
get ciphers;
```

get class-of-service Command—Display the CoS Templates in Use

The *get class-of-service* command displays the class-of-service templates that are in use in a DSA. Each class-of-service template in use is listed with the template label at the top.

Class-of-service templates are deprecated. Use views instead.

This command has the following format:

```
get class-of-service;
```

For example, the template for the standard class of service at Excellent ISP would appear like this:

```
************** standard **************
class-of-service =
target object class : excellentISPUser
target attribute : excellentISPPackage
target value : "Standard"
attribute list =
attribute : excellentISPmailQuotaMB
value/s : "20"
disposition : default
attribute : excellentISPwebSpaceMB
value/s : "20"
disposition : default
attribute : excellentISPaccessHours
value/s : "15"
disposition : override
attribute : excellentISPprice
value/s : "19.95"
disposition : default
attribute : excellentISPextraHoursPrice
value/s : "1.00"
disposition : default
```

get dsa Command—Display Information About the DSA

Use the *get dsa* command to view information about a particular DSA.

This command has the following format:

```
get dsa dsaname;
```

get dsas Command—Display All Known Remote DSAs

Use the *get dsas* command to view information about all known remote DSAs.

This command has the following format:

```
get dsas;
```

get dsp Command—Display the DSP Configuration Values

The *get dsp* command shows the current DSP configuration values of the DSA.

This command has the following format:

```
get dsp;
```

get dynamic-group Command—Display Any Dynamic Groups

This command has the following format:

```
get dynamic-group;
```

get help Command—Display the DXserver Console Commands

The *get help* command displays useful DXserver console commands.

This command has the following format:

```
get help;
```

get history Command—Show How Many Console Commands the history Command Will Display

The *get history* command tells you how many previously entered console commands the system will store and display when you enter the *history* command, that is, how many commands will be remembered.

This command has the following format:

```
get history;
```

To change the number of previously entered commands the system will display, see set history command.

More information

history Command—Display Previously Entered Console Commands (see page 45)
set history Command—Set the Number of Console Commands the history Command Will Display (see page 78)

get log Command—List All Log Files

This command has the following format:

```
get log;
```

get mib Command

This command has the following format:

```
get mib;
```

get online-dsa Command—Display the Configuration of a DSA

The *get online-dsa* command shows the knowledge of a particular DSA, if it is online.

Note: This command is only useful for DSAs that use the pre-r12 configuration.

This command has the following format:

```
get online-dsa dsa-name;
```

get online-dsas Command—Display Current Connections to a DSA

The *get online-dsas* command displays information about current outgoing connections to other DSAs.

This includes both remote and local DSAs. To see a list of remote DSAs, use the command *get dsas*.

This command has the following format:

```
get online-dsas;
```

get opensslversion Command—Display the OpenSSL Version

The *get* opensslversion command displays the OpenSSL version CA Directory uses.

This command has the following format:

```
get opensslversion;
```

get oper Command—Display Operational Settings

The *get oper* command displays the operational settings.

This command has the following format:

```
get oper;
```

Example: Output from the *get oper* Command

```
max-local-ops     = 200
max-op-time       = 120
max-op-size       = 50
access-controls   = TRUE
op-attrs          = TRUE
read-only         = FALSE
prune-oc-parents  = FALSE
return-oc-parents = FALSE
add-oc-parents    = FALSE
modify-on-add     = FALSE
ignore-name-bindings = FALSE
alias-integrity   = TRUE
relaxed-not-search = FALSE
timestamp-resolution-seconds = FALSE
password-storage  = sha-1
```

get referential-integrity Command—Display Referential Integrity Rules

Use the *get referential-integrity* command to view all the referential integrity rules.

This command has the following format:

```
get referential-integrity;
```

get schema Command—Display the Schema

This command has the following format:

```
get schema for identifier;
```

identifier

A name or an object identifier of any schema definition (an attribute, object class, name binding, prefix, or attribute set).

Example: Retrieve the Definition of *commonName*

Use **one** of the following commands:

```
get schema for commonName;
get schema for (2.5.4.3);
```

An example output of this command is:

```
Attribute (2.5.4.3)
    name = commonName
    ldap-names = cn
    syntax = caseIgnoreString
```

get stack Command—Display Communication Settings, Such as Port Numbers

Use the get stack command to display the communication settings in a DSA. The command has the following format:

```
get stack;
```

Example Output from the get stack Command

```
dap-psap         = ""
dsp-psap         = ""
disp-psap        = "DISP"

address          = 207.2.6.99 port 19389
snmp-port        = 19389
console-port     = 19390
snmp-description = CA eTrust DXserver
snmp-contact     = supportconnect@ca.com
snmp-name        = democorp
snmp-location    = http://www.ca.com
```

get stats Command—Display Operational Statistics

The *get stats* command returns a list of operation statistics.

This command has the following format:

```
get stats;
```

get trace Command—Display the Trace Settings

The *get trace* command displays the current trace settings for this DSA.

This command has the following format:

```
get trace;
```

get trace-level Command—Display the Trace Level

This command displays the current trace level.

It has the following format:

```
get trace-level;
```

get user Command—Display the DSA Configuration

The *get user* command displays a list of DSA configuration items.

This command has the following format:

```
get user;
```

get users Command—Display the Current Bindings

The *get users* command displays a list of current incoming and outgoing bindings.

This command has the following format:

```
get users;
```

More information:

abandon Command—Cancel Operations (see page 27)

get user-threads Command—Display Thread Settings

The *get user-threads* command displays the number of user threads.

This command has the following syntax:

```
get user-threads;
```

get view Command—Display View Definitions

Use the *get view* command to display the view definitions in the DSA.

This command has the following format:

```
get view;
```

help Command—Display Help on a Command

The help command lists the commands for the DSA administration modules, X.500 services, and management scripts.

This command has the following format:

```
help;
```

Example: Use the *help* Command

When you are not sure of the syntax, try a nonsense word for the missing part, or leave the command incomplete and let the resulting error message tell you what is expected:

```
>>>> oper get fred;
-----------------^

Syntax Error: Expecting “config" or stats".
```

Example: Output of the *help* Command

When you enter the help command at the console, the response is similar to:

```
DXserver rXX.X (build X) Platform
Please consult the Administration Guide for the full set of supported commands.
Some useful DXconsole commands:
set trace =     alert, cert, connect, dsa, diag, error, query, ldap
                limit, warn, stack, stats, summary, time, update, x500
get (general)   dsas, dsp, history, log, mib, online-dsas, oper,
                schema for <item>, stack, stats, trace, user, users,  user-threads
get (specific) access, allow-search, agreement <id>, agreements, cache,
                class-of-service, dynamic-group, referential-integrity, view
```

history Command—Display Previously Entered Console Commands

The *history* command displays previously entered console commands. You can use this to simply view the previously entered console commands, or you can re-use the commands.

This command has the following format:

```
history;
```

You can also shorten this command to:

```
h;
```

Each command is displayed with a unique consecutive number. To reuse a command, enter the number, followed by a semi-colon, that corresponds to the command you want to use. For example:

```
10;
```

At any time you can reuse the last command you entered, by entering:

```
0;
```

For information on how to set the number of previous commands this history command will display, see *set history*.

Example: Get the history and reuse a previous command

The following example shows how to use the set history and history commands. In this example, the user sets the number of items returned by the history command to 5, then gets the history, and then reuses the *get cache* command. The user input is indicated by bold text.

```
dsa> set history = 5;
dsa> history;
dsa>     10 get dsp;
dsa>     11 get log;
dsa>     12 get cache;
dsa>     13 bind-req;
dsa>     14 search-req base-object = <c AU>;
dsa> 12;
```

More information:

get history Command—Show How Many Console Commands the history Command Will Display (see page 39)
set history Command—Set the Number of Console Commands the history Command Will Display (see page 78)

logout Command—Close the DSA Console

The logout command closes the DSA console telnet session. This is the usual way to terminate a DSA console session.

This command has the following format:

```
logout;
```

reset class-of-service Command

This command has the following format:

```
reset class-of-service;
```

reset stats Command

This command has the following format:

```
reset stats;
```

set access-controls Command—Set Access Controls

This command enables or disables access controls.

This command has the following format:

```
set access-controls = true | false;
```

More information:

set reg-user Command—Configure Registered User Access Level Rights (see page 122)
set public-user Command—Configure Anonymous User Access Level Rights (see page 116)
set admin-user Command—Configure Administrative User Access Level Rights (see page 48)
set super-user Command—Configure Super User Access Level Rights (see page 130)

set add-oc-parents Command

When set to *true*, this command causes DXserver to add superior object classes even if the client did not specify these while adding an entry.

It complements the *prune-oc-parents* and *return-oc-parents* flags and is added for Netscape compatibility.

This command has the following format:

```
set add-oc-parents = true | false;
```

set admin-user Command—Configure Administrative User Access Level Rights

This command grants specified access rights at the administrative user access level, to specified users, over a specified scope.

Access rights granted at this access level cannot be taken away by other access control rules.

Administrative user access controls rules are effective only when you enable access controls.

This command has the following format:

```
set admin-user [tag] = {
    users
    scope
    [attrs      = attribute-list]
    [perms      = permission-list]
    [auth-level = simple | ssl-auth]
    [validity   = [start hhmm end hhmm] [on day]]
};
```

tag

(Optional) Defines a name for this rule.

users

Defines the users that this rule applies to, where *users* is one of the following:

user = *DN*

Defines the user that this rule applies to.

role = *DN*

Defines the role that this rule applies to.

group = *group-name*

Defines the access control group that this rule applies to. Use of access control groups is deprecated, so use of this option is also deprecated.

user-subtree = *DN*

Defines the top of the subtree of users that this rule applies to.

own-entry

Specifies that the users defined in *scope* have access to their own entries only.

own-subtree

Specifies that the users defined in *scope* have access to their own entries and any entries below their own entry.

scope

Defines the area of the DIT that this rule gives access to, where *scope* is one of the following:

entry = *DN*

Specifies the entry that this rule grants or denies access to.

subtree = *DN*

Specifies the subtree that this rule grants or denies access to.

attrs = *attribute-list*

(Optional) Defines the attributes or attribute set to which this rule applies, where *attribute-list* is a comma-separated list of attribute names.

If *attrs* is not specified, then the access rule applies to the whole entry. *add* and *remove* permissions require that *attrs* is not specified.

perms = *permission-list*

(Optional) Specifies the permissions that this rule grants to the *users* for the *scope*.

If *perms* is not specified, then all permissions are granted.

permission-list is a comma-separated list of one or more of the following:

all

Specifies that users have all available permissions over the scope. This option implies all of the permissions listed below.

read

Specifies that users can read the information defined in the scope.

add

Specifies that users can add to the information defined in the scope.

remove

Specifies that users can delete entries defined in the scope.

modify

Specifies that users can change information defined in the scope.

rename

Specifies that users can rename the entries defined in the scope.

auth-level = simple | ssl-auth

(Optional) Specifies the level of authentication required. If you use this option, use one of the following:

simple

Specifies that this rule only applies to users that bind using simple authentication (username and password).

ssl-auth

Specifies that this rule only applies to users that bind using SSL authentication.

validity = [start *hhmm* end *hhmm*] [on *day*]

(Optional) Defines the period during which this rule is valid. Use any of the following:

start *hhmm* end *hhmm*

Defines the start and end of the period during which this rule is valid.

on *day*

Defines the day on which this rule is valid, where *day* is a string like 12345 or 67 (1 is Monday).

Example: Give Administrative Access to All Users in a Subtree

The following command gives all users in the Finance subtree access to the Corporate subtree:

```
set admin-user Finance-Users" = {
 user-subtree = <c AU"><o Democorp"><ou Corporate"><ou Finance">
 subtree = <c AU"><o Democorp"><ou Corporate">

};
```

Example: Give Administrative Privileges to a Role

The command in this example gives users in the role *project-leader-group* read and update privileges to the *Technology SIG* entry if they bind to the DSA using SSL authentication:

```
set admin-user project-leaders" = {
 role    = <c AU"><o Democorp"><ou roles"><cn project-leader-group">
 entry   = <c AU"><o Democorp"><ou R&D"><listName Technology SIG">
 auth-level      = ssl-auth

};
```

Example: Let Users Update a Single Attribute

The command in this example lets all users in the group *pabx-mgmt-group* update the attribute *workPhone* in the R&D subtree:

```
set admin-user work-phone" = {
 group   = pabx-mgmt-group"
 subtree= <c AU"><o Democorp"><ou R&D">
 attrs   = workPhone
};
```

Example: Let Users Update Some Attributes in Their Own Entry

The command in this example gives all users in the R&D subtree permission to update the values of the attributes *workPhone* and *description* in their own entry only:

```
set admin-user my-own-work-details" = {
 own-entry
 subtree= <c AU"><o Democorp"><ou R&D">
 attrs   = workPhone, description
};
```

set agreement Command—Create a DISP Agreement

This command has the following format:

```
set agreement id.version =
{
        initiator       = dsaname { supplier | consumer }
        responder       = dsaname { anonymous | clear-password | ssl-auth }
        [relay          = dsaname { anonymous | clear-password | ssl-auth }]
        area            = DN
        [filter = search-filter ]
        [attributes     = attribute-list ]
        [strategy       = onchange | frequency time type ]
};
```

id.version

The agreement identification and version numbers (for example, 1.2).

initiator

The DSA initiating the DISP update:

dsaname

The name of the initiating DSA.

supplier

Specifies that the initiating DSA is acting as a supplier.

consumer

Specifies that the initiating DSA is acting as a consumer.

responder

The DSA receiving the DISP update.

dsaname

The name of the responding DSA.

anonymous

Specifies that the responding DSA can receive an anonymous connection.

clear-password

Specifies that the responding DSA requires a user ID and password.

ssl-auth

Specifies that the responding DSA requires a SSL authentication.

relay

(Optional) Specifies an intermediate DSA used to relay the DISP update:

dsaname

Names the relay DSA.

anonymous

Specifies that the relay DSA can receive an anonymous connection.

clear-password

Specifies that the relay DSA requires a user ID and password.

ssl-auth

Specifies that the relay DSA requires a SSL authentication.

area = *DN*

Specifies the distinguished name of the entry at the top of the subtree covered by this agreement.

filter = *search-filter*

(Optional) Specifies an X500 search filter that restricts the entries to be included in the DISP update.

attributes = *attribute-list*

(Optional) Specifies a comma-separated list of attributes to include or exclude from the DISP update.

strategy

(Optional) Enables automatic DISP updates, or specifies when the updates will take place:

onchange

Enables automatic DISP updates.

frequency

Specifies the update frequency. Use one of the following values: *hourly*, *daily*, *weekly*, or *monthly*.

time

Specifies the time at which the update will occur. Use one of these formats: dd:hh:mm or hh:mm.

type

Specifies the type of update. Use one of the following values: *incremental* or *full*.

Example: Create a DISP Agreement

```
set agreement 0.0 =
{
    initiator = EAGLE supplier
    responder = BACKUP anonymous
    area      = <c AU><o Democorp>
    strategy  = daily 03:00 incremental
};
```

Example: Create a Selective Shadowing Agreement

```
set agreement 2.1 =
{
    initiator = EAGLE supplier
    responder = BACKUP anonymous
    area = <c AU><o Democorp>
    filter = { attr = objectClass value = organizationalPerson }
    attributes = { { exclude = title, description, telephoneNumber } }
    strategy = on-change
};
```

set alias-integrity Command—Set the DSA to Manage the Integrity of Aliases

The *set alias-integrity* command specifies whether the DSA manages the integrity of alias entries, for example, deleting aliases that point to deleted entries.

This command has the following format:

```
set alias-integrity = true | false;
```

set allow-binds Command—Enable or Disable New Bindings on a DSA

The *set allow-binds* command specifies whether the DSA is accepting new binding requests.

This command has the following format:

```
set allow-binds = true | false;
```

true

New bindings are accepted. This is normal operational behavior.

false

New bindings are rejected. Use this option as a prelude to performing a graceful shutdown.

set allow-native-prefix-reauthentication Command—Allow Router DSAs to Use a Prefix-mapped User Name to Authenticate

The *set allow-native-prefix-reauthentication* command allows a router DSA to use a prefix-mapped user name to authenticate to this DSA.

This is useful in the following situtation: If DSA A and DSA B both have a native prefix and if the router has accessed DSA A on behalf of a client application, and this same client application tries to access DSA B, and if in addition the router cannot bind to DSA B because the username apparently doesn't address an entry in DSA B but the previously prefix-mapped username from the access to DSA A is apparently in the non-prefix-mapped DSA B, the router will use this prefix-mapped username to authenticate to DSA B.

This command has the following format:

```
set allow-native-prefix-reauthentication = <bool>;
```

set allow-search Command—Define a Search Profile

The *set allow-search* command defines a search profile to the DSA. You use search profiles to restrict the searches that users can perform.

This command has the following format:

```
set allow-search profileName = {
    (
        scope = allowedScope [,allowedScope ...]
        [filter = allowedFilter [,allowedfilter ...]]
    )
    [,(
        scope = allowedScope [,allowedScope...]
        [filter = allowedFilter [,allowedfilter...]]
    )]
    ...
};
```

profileName

Defines the name that the DSA command interpreter uses to identify the search profile. If the name contains spaces or non-alphanumeric characters, then it must be enclosed in quotes.

scope = *allowedScope [,allowedScope ...]*

Specifies a search scope that this search profile will allow. A scope parameter can allow multiple scope specifications, which will all share the same filter specification.

allowedScope is one of the following:

read

Allows base-object searches.

browse

Allows one level searches.

subtree

Allows whole subtree searches.

all

Allows searches of any scope.

filter = *allowedFilter [,allowedFilter ...]*

(Optional) Specifies the filters that are allowed for searches with the preceding search scope. Each scope parameter can have multiple associated filters.

allowedFilter is one of the following:

and

Allows the AND operator in filters, for example, (&(oc=inetOrgPerson)(cn="john smith")).

any

Allows filters with wildcards anywhere, for example, (cn=*john*smith*).

approx

Allows pattern/phonetic matching, for example, (cn~=john smith).

equality

Allows exact match filter items, for example, (cn=john smith).

final

Allows filters with a leading wildcard, for example, (cn=*smith).

greater-or-equal

Allows the >= operator in filters, for example, (retries>=3).

initial

Allows filters with a trailing wildcard, for example, (cn=john*).

less-or-equal

Allows the <= operator in filters, for example, (retries<=3).

none

Allows searches that do not contain a filter.

not

Allows the NOT operator in filters, for example, (!(cn="john smith")).

or

Allows the OR operator in filters, for example, (|(cn="john doe")(cn="john smith")).

present

Allows filters for presence, for example, (objectClass=*).

Example: Allow Any Search

```
set allow-search superusers = {
    ( scope = all )
    };
```

This search profile allows any search.

Example: Allow Base-Object Searches Only

```
set allow-search Guest = {
    ( scope = read )
    };
```

This search profile allows base-object searches only. No filter is specified with the scope parameter, so the search can contain any filter or none.

Example: Allow Base-Object Searches and Some Other Searches

```
set allow-search userDefault = {
    (scope = read),
    (scope = browse, subtree
     filter = and, or, equality)
    };
```

This search profile allows base-object searches with any filter and other searches if they have a filter for exact matches.

set allow-search-default Command—Make One Search Profile the Default

The *set allow-search-default* command sets a search profile to be the DSA's default search profile.

This command has the following format:

```
set allow-search-default = profileName;
```

profileName

Specifies the profile to be used as the default search profile. The default search profile is applied if, and only if, a user does not have a role that defines a search profile.

set attr-set Command—Define an Attribute Set

The *set attr-set* command defines an attribute set.

An attribute set consists of an attribute set name and a list of previously defined attributes or attribute sets.

This command has the following format:

```
set attr-set <sSetAttrSet>;
```

set attribute Command—Define an Attribute

The *set attribute* command defines an attribute. An attribute is a basic building block in the schema, and it consists of an attribute name, alternate name, syntax, single-valued flag, and a description.

For more information, see Manage Schemas.

This command has the following format:

```
set attribute OID[:OID_Suffix] = {
        name            = attribute-name
        ldap-names      = cn
        equality        =
        ordering        =
        substr          =
        syntax          =
        [ single-valued | multi-valued ]
        [ no-user-modification ]
        [ description ]
};
```

OID

Specifies the object ID for the attribute. An OID has one of the following forms:

- A dotted numeric string surrounded by brackets (), for example:

  `(2.5.4.10)`

- The name of an OID, as produced by the set OID-Prefix command

OID_Suffix

(Optional) Further specifies the OID.

name = *attribute-name*

Specifies the name of the attribute. This is its formal name, and is often descriptive.

ldap-names

Specifies alternative names for the attribute. These are similar to nicknames: they can be used anywhere the name can be used. Often these are shorter, and used in DNs. For example, *c* for country.

equality

Indicates the type of matching to apply to the attribute.

ordering

Indicates the ordering rules to apply to the attribute.

substr

Indicates the substring-matching rule to apply to the attribute.

syntax

Indicates the type of data that may be stored in the attribute.

single-valued | multi-valued

Indicates whether the attribute has multiple values (for example, lines of an address), or is limited to a single value (for example, salary).

no-user-modification

Indicates whether the user can modify the value of the attribute

description

A description of the attribute.

Example: *set attribute* Command

```
set attribute (2.5.4):10 = {
name = commonName
ldap-names = cn

syntax = caseIgnoreString ;
```

set auth-trap Command—Set the DSA to Raise an SNMP Trap When Authentication Fails

The *set auth-trap* command is a mechanism that can be used to hook into the authentication processing on the DSA using SNMP traps. When set to *true*, this command turns on raising an authentication trap whenever an authentication failure occurs.

This command has the following format:

```
set auth-trap = true | false;
```

set busy-for-referral Command—Send Busy in Place of LDAP Referrals

When busy-for-referral is true, the DSA does not send LDAP referrals to clients. Instead, it sends the error "busy". The default is false.

This command has the following format:

```
set busy-for-referral = {true | false} ;
```

set cache-index Command—Specify Attributes to Be Indexed

This command defines which attributes will be indexed.

The cache responds to a search filter that uses one of these attributes. You can define as many as you like, separated by commas. Memory requirements are affected.

This command has the following format:

```
set cache-index = attribute-list | all-attributes;
```

set cache-index-all-except Command—Specify Attributes Not to Index

This new command defines which attributes will *not* be indexed. This lets you reduce the amount of indexes that are loaded into memory.

This command behaves like cache-index-all except that it also allows you to specify a number of attributes that will not be indexed. It may be simpler to specify all except" than to give a complete list of attributes to index, as for cache-index, especially as trying to specify all attributes to index may not cover new attributes that are added in the future.

It may be necessary to limit the number of attributes that are indexed as indexing an attribute requires additional memory. If indexing all attributes would consume all the memory allocated to DXgrid, this command could help tune the memory requirements.

It is a good idea to index attributes, because searches require indexes to evaluate their filters. Also, indexes are required to find specific entries that are targeted by updates. On the other hand, searches may still proceed even where few attributes are indexed, but they may take much longer to process. There is therefore a trade-off between memory and performance. Where memory is not a problem it would be best to index all attributes.

You can define as many attributes as you like, separated by commas.

This command has the following format:

```
set cache-index-all-except = attribute-list;
```

set cache-reverse Command—Specify Which Cached Attributes to Reverse-index

This command has the following format:

```
set cache-reverse = attribute-list | all-attributes;
```

set check-structural-oc—Prevent Entries with Multiple Unrelated Structural Object Classes from Being Created

By default, CA Directory lets you create an entry with multiple structural object classes that are not part of a single inheritance chain. While this ability can be useful, it does not conform with Section 8.3.2 of X.501 and Section 2.4.2 of RFC 451.

If you do not want entries with multiple unrelated structural object classes to be created, you can set a schema flag to enforce this.

This command has the following format:

```
set check-structural-oc = true|false;
```

If *set check-structural-oc* is set to *true*, it will not be possible to add an entry which has more than one structural object class hierarchy.

set class-of-service Command—Create a Class-of-Service Template

Class of service templates are deprecated. Use Views instead.

This command has the following format:

```
set class-of-service <cosLabel> = { <operCos> };
```

set concurrent-bind-user Command—Allow the DSA to Process Concurrent Binds

The *set concurrent-bind-user* command allows a DSA to process concurrent binds. This is necessary when you use a CA Directory DSA with SiteMinder.

In order that SiteMinder process authentications asynchronously, you must set the DSA to do both of the following:

- Pretend to be a Netscape server to Siteminder (see set mimic-netscape-for-siteminder Command (see page 85))
- Process concurrent binds

Any DSA that is needed for authenticating concurrent binds needs the rebind link-flag set in its knowledge.

This command has the following format:

```
set concurrent-bind-user = DN;
```

DN

The username that SiteMinder uses to bind to the directory.

set connect-log Command—Record Each Connection in the Connect Log

The *set connect-log* command opens the connection log.

When a connection log is open, a line for each connection made when the connection is released is added to the connection log file.

Writing to the connect log is not affected by the trace setting. It is time- and date-stamped, and a new one is written daily.

If the log file already exists, the DSA adds the new output to the existing log file. If the log file does not exist, it is created.

This command has the following format:

```
set connect-log = log-name;
```

log-name

Specifies the name of the file. If the log file name contains the string $S then the system substitutes the name of the DSA. For example, if the DSA name is *democorp*, the log-name */logs/warn-$S* specifies the file name *warn-democorp_YYYYMMDD.log*.

set credits Command—Limit the Number of Operations per User

The *set credits* command regulates the DSA. New requests are accepted until the number of credits is exhausted.

This command has the following format:

```
set credits = number-credits;
```

number-credits

Specifies the maximum number of concurrent operations the DSA processes for each user binding.

set dereference-alias-on-bind Command—Follow Alias on Bind Request

The *set dereference-alias-on-bind* command allows the DXserver to follow aliases when processing binds. By default the DXserver will not follow aliases when processing bind requests.

This command has the following format:

```
set dereference-alias-on-bind = true|false
```

set disable-client-binds Command—Set the DSA to Refuse Client Binds

The *set disable-client-binds* command sets the DSA to refuse any binds from a DAP or LDAP client, with the message *unavailable*. Use this command to force clients to go through the router DSA.

This command has the following format:

```
set disable-client-binds = true | false;
```

set disable-transaction-log—Disable or Enable the Transaction Log

The *set disable-transaction-log* disables or enables the transaction log. Disabling the log provides faster performance on writes, but also prevents recovery (unless recovery is provided by replication).

This command has the following format:

```
set disable-transaction-log = true | false;
```

The default is *false*.

set disable-transaction-log-flush—Disable or Enable Transaction Log Flushing

The *set disable-transaction-log-flush* disables or enables transaction log flushing. Flushing the transaction log reduces performance on writes (down to ~100 updates per second). Without flushing, transaction logging performs much better (at ~10,000 updates per second). However, flushing allows you to restart the DSA after an abnormal termination or power failure, avoiding disaster recovery procedures.

This command has the following format:

```
set disable-transaction-log-flush = true | false;
```

The default is *false*.

set dsa Command—Define the Knowledge Settings of a DSA

If you do not use DXmanager, use the *set dsa* command to define the knowledge of a DSA.

Important! You must declare the parameters in the order shown.

This command has the following format:

```
set dsa dsaname =
{
    prefix              = DN
    [ native-prefix     = DN ]
    dsa-name            = DN
    [ dsa-password      = password ]
    [ ldap-dsa-name     = DN ]
    [ ldap-dsa-password = password ]
    address             = tcp hostname port port-number [ ,tcp hostname2 port
    port-number2 ]
    [ tsap              = tsel ]
    [ ssap              = ssel ]
    [ osi-psap          = psel ]
    [ disp-psap         = dispsap ]
    [ cmip-psap = cmipsap ]
    [ snmp-port         = port-number ]
    [ console-port      = port-number ]
    [ remote-console-port       = port-number ]
    [ remote-console-ssl        = true | false ]
    [ console-password  = password | "{password-format}password-hash" ]
    [ auth-levels       = anonymous | clear-password | ssl-auth ]
    [ dsp-idle-time     = idle-time ]
    [ dsa-flags = dsaflag-list ]
    [ trust-flags       = trustflag-list ]
    [ link-flags        = linkflag-list ]
};
```

dsaname

Specifies the name of the DSA.

prefix

Specifies a partial DN, which specifies the namespace partition served by this DSA.

native-prefix

Specifies a partial DN, which the DSA recognizes as applicable to its entries. This is generally only used with LDAP servers.

dsa-name

Specifies the name of the DSA as a DN; not to be confused with the name of the server

dsa-password

Specifies the password other DSAs must supply to communicate with this DSA.

ldap-dsa-name

Specifies the name of the LDAP DSA.

ldap-dsa-password

Specifies the password of the LDAP DSA.

address

Specifies one or more TCP/IP addresses for the DSA in one of the following forms:

- address = tcp "*IP address*" port *number*
- address = tcp "*host name*" port *number*

If there is a choice of addresses associated with the host name, the IPv6 address is selected. To specify the IPv4 address, replace the string **tcp** with **ipv4**. To specify the IPv6 address, replace the string **tcp** with **ipv6**.

Note: The SNMP trap address remains at IPv4.

Example: Specifying an IP address for IPv4 environments:

```
address = tcp "345.785.987.224" port 19389
```

Example: Specifying an IP address for IPv6 environments:

```
address = tcp "fe80::20d:56ff:fed4:8300%5" port 19389
```

Example: Specifying an IP address for hybrid IPv6/IPv4 environments:

```
address = tcp "fe80::20d:56ff:fed4:8300%5" port 19389, tcp "345.785.987.224" port 19389
```

Example: Specifying a host name:

```
address = tcp "eagle" port 19389
```

Example: Specifying a host name for IPv4 environments:

```
address = ipv4 "eagle" port 19389
```

tsap

Specifies a Transport SAP port number. This is not often used.

ssap

Specifies a Session SAP port number. This is not often used.

osi-psap

Specifies a Presentation SAP port number. This is not often used.

disp-psap

Specifies DISP Presentation SAP. If this is not set, DISP is disabled.

cmip-psap

CMIP is no longer supported.

snmp-port

Specifies the SNMP port.

console-port

Specifies the console port address, which allows the DSA console to accept connections from the local computer. If this is not specified, the DSA does not have a local console.

remote-console-port

Allows the DSA console to accept a connection from a remote computer on this port. When this is not specified, there is no remote console for the DSA.

remote-console-ssl

Forces the DSA to encrypt console sessions when it runs remotely.

console-password

The password required for connections from a remote computer. This password is transmitted in clear text.

auth-levels

Specifies the levels of authentication that will be accepted by this DSA. May include *anonymous*, *clear-password*, and *ssl-auth*.

dsp-idle-time

Specifies the maximum time (in seconds) that a DSP connection can be idle before it is disconnected.

dsa-flags

Specifies the flags that control the operation of the DSA. DSA flags are as follows:

limit-list

Disables the list operation on the DSA.

limit-search

Restricts complex searches or searches with no filter on the DSA.

limit-search-exact

Limits a DSA to performing exact searches, that is searches with a single equality filter item with no wildcards.

load-share

Marks a DSA as part of a load share group. The DSA should have other peer DSAs with the same prefix, which are also marked as load-share. A router DSA shares operations over each DSA in the load share group.

multi-write

Marks a DSA as part of a multiwrite group. The DSA should have other peer DSAs, with the same prefix, which are also marked as multiwrite. Updates are automatically propagated to all peer DSAs marked as multiwrite.

multi-write-async

Makes the DSA update asynchronously, even though it is in a multiwrite group.

multi-write-group-hub

Specifies which DSAs in the group acts as the hub. This only works if you also have multi-write-group enabled. This setting prevents unsuitable DSAs being selected as the hub in a failover situation.

no-routing-ac

Permits forwarding of a request to another DSA regardless of access control constraints.

no-service-while-recovering

While this DSA is in recovery mode, it only accepts updates from peers: this prevents clients from accessing out-of-date data.

read-only

Disables update operations on the DSA.

relay

Permits a router DSA to exist without consuming a level of the DIT.

shadow

Permits a DSA to be updated by DISP or multiwrite, but prevents any other updates, for example, through DAP or LDAP.

trust-flags

Specifies flags relating to trust that control the operation of the DSA. Thrust flags are as follows:

allow-check-password

Permits a DSA, while processing a bind request from a user who is not local, to pass a name and password-compare request to this DSA. The result of the compare request is then used to authenticate the user.

trust-conveyed-originator

Signifies that a DSA treats the originator and authentication level passed in DSP chaining arguments as if that user and authentication level were authenticated locally.

allow-upgrading

Lets the DSA pass an anonymous user request across an authenticated DSP link.

allow-downgrading

Lets the DSA pass an authenticated user request across an anonymous DSP link.

no-server-credentials

Removes the requirement for mutual authentication and permits a link to be set up if the remote DSA does not send credentials in the bind response.

link-flags

Specifies flags that control connecting to the DSA. Link flags are as follows

dsp-ldap

The DSA is treated as an LDAP server that supports LDAP 3.0. Other DSAs will send requests to the DSA as if it was an LDAP server.

When dsp-ldap is configured, there will be no COMPARE operation on the userPassword attribute, following a bind. If the same user connects more than once, that user will use the same link, and dxserver will check that the user and the password are the same.

dsp-ldap-proxy

Causes the last DSA in the chain to use the authorization of the originating user to perform operations on the LDAP server.

dsp-ldapv3

The DSA is treated as an LDAP server that supports LDAP 3.0.

ms-ad

The DSA is treated as an Active Directory service. If you observe any problems with linking to Active Directory, set this flag.

ms-exchange

The DSA is treated as a Microsoft Exchange server to overcome limitations in Exchange's version of LDAP.

nexor

Allows this DSA to bind anonymously to a Nexor DSA. To bind anonymously with a Nexor DSA, the message ID must be stripped of all identifying credentials.

rebind

Allows this DSA to support concurrent binds. If this flag is not set on a link that a DSA requires for authenticating concurrent binds, these binds will fail. Used in conjunction with the set concurrent-bind-user command.

Note: Only use this flag for LDAP directories. If you do not use dsp-ldap, we recommend that you do not use use rebind either.

siemens

Allows this DSA to bind anonymously to a Siemens DSA. To bind anonymously with a Siemens DSA, the message ID must be non-zero.

ssl-encryption

All DSA-to-DSA communication to the DSA with this link flag uses SSL encryption.

ssl-encryption-remote

It is similar to ssl-encryption, but SSL encryption is not used if the target DXserver is on the same host.

unavailable

Marks a DSA as unavailable. A DSA will not forward requests to a DSA marked as unavailable.

set dsa prefix—Define the Prefix of a DSA

The prefix parameter of the set DSA command defines the prefix for a DSA. You can include a definition of a horizontal partition in the prefix.

The syntax of the prefix parameter is as follows:

`prefix = x500DN [ horizontal_partition ]`

X500DN

Defines a partial DN in X.500 format.

horizontal_partition

Defines a horizontal partitioning for a DSA and has the following syntax:

`< horizontal_partition_attribute "[hashn(total )= DSAid ]">`

The symbols in the above line (< " [( ) =] " >) are part of the command.

horizontal_partition_attribute

Defines the RDN that specifies the entry, for example UserID.

hashn

Specifies the hash algorithm that the router will use to partition the namespace. This can be one of the following:

hash1

Specifies that the router uses SHA-1 hash

hash2

Specifies that the router sums the ASCII values of the uppercase characters as the hash

total

An integer that defines the total number of DSAs in the horizontal partition.

DSAid

An integer from zero to *total-1* that defines the DSA. For example if *total* is 3, *DSAid* is 0, 1, or 2.

set dxconsole-connect-alert Command—Set Tracing for Console Connections

Corporate policy may require audit information for users that connect to the DSA console. This is because by using the console a user can bypass any configuration version control systems that may be in place and change configuration options.

If dxconsole-connect alert is enabled, when a user successfully connects to the DSA console, the system writes a message containing the IP address and username (if 'dxconsole-users' is set) to the trace log.

This command has the following format:

```
set dxconsole-connect-alert = {true | false};
```

set dxconsole-users Command—Specify Which Users Can Connect to the DSA Console

The *set dxconsole-users* command allows users to connect to the DSA console by logging in with their DN and password.

This command has the following format:

```
set dxconsole-users = [users], [roles];
```

users

Specifies the users who can connect to the console for this DSA, as a comma-separated list of DNs.

roles

Specifies the roles which can connect to the console for this DSA, as a comma-separated list of DNs.

Example: Craig Link Can Connect to the Democorp Console

The following command allows Craig Link to connect to the Democorp console:

```
set dxconsole-users = <c AU><o Democorp><ou ><ou Corporate><ou Administration><cn
"Craig Link>;
```

Now, Craig Link can connect to the console by logging in with the following details:

- cn=Craig Link,ou=Administration,ou=Corporate,o=Democorp,c=AU
- Craig Link's password

Example: Craig Link and the Manager Role Can Connect to the Democorp Console

The following command allows Craig Link *and* all users with the Manager role to connect to the Democorp console:

```
set dxconsole-users = <c AU><o Democorp><ou ><ou Corporate><ou Administration><cn
"Craig Link>,<c AU><o Democorp><ou Roles><cn ConsoleUsers>;
```

set dxgrid-backup-location—Define the Backup Location

This command defines the backup location of the datastore (.db) file.

This command has the following format:

```
set dxgrid-backup-location = pathfilename
```

pathfilename

Defines the path and filename of the backup files. If the path is relative, it is relative to DXHOME.

Example: Defining a Relative Path for a Datastore

```
set dxgrid-backup-location = backup
```

set dxgrid-db-location Command—Define the Path to the Datastore

This command defines the location of the datastore.

This command has the following format:

```
set dxgrid-db-location = pathfilename
```

pathfilename

Defines the path and filename of the datastore. If the path is relative, it is relative to DXHOME.

Example: Defining a Relative Path for a Datastore

```
set dxgrid-db-location = data
```

set dxgrid-db-size Command—Define the Size of the Datastore

This command defines the size of the datastore. One occurrence of this command must be in the initialization configuration file.

This command has the following format:

```
set dxgrid-db-size = sizeinMB
```

sizeinMB

Defines the size of the datastore in MB.

set dxgrid-tx-location—Define the Transactions Files Location

This command defines the location of the transaction files.

This command has the following format:

```
set dxgrid-tx-location = pathfilename
```

pathfilename

Defines the path and filename of the transaction files. If the path is relative, it is relative to DXHOME.

Example: Defining a Relative Path for a Datastore

```
set dxgrid-tx-location = transaction_files
```

set dxgrid-queue–Add a Queue in Front of Data Store

This command adds a queue in front of the data store.

This command has the following format:

```
set dxgrid-queue=true
```

True

Changes the scheduling behaviour of DXgrid. This can be helpful in some scenarios. For example updates should be returned continuously so that the response time for each update should be small.

set dynamic-group Command

This command has the following format:

```
set dynamic-group [tag] = {
    objectclass = object-class
    url-attr = attribute
    member-attr = attribute

};
```

objectclass = *object-class*

Specifies that dynamic groups use the *groupOfURLs* object class.

url-attr = *attribute*

Specifies that the *memberURL* attribute stores the LDAP filter that defines the group membership.

member-attr = *attribute*

Specifies that the result of the LDAP filter in the *memberURL* attribute will be stored in the *member* attribute.

set force-encrypt-anon Command—Force Users to Use SSL on Anonymous Binds

The *set force-encrypt-anon* command forces SSL encryption on anonymous binds.

This command has the following format:

```
set force-encrypt-anon = true | false;
```

set force-encrypt-auth Command—Force Users to Use SSL on Authenticated Binds

The *set force-encrypt-auth* command forces SSL encryption on authenticated binds.

This command has the following format:

```
set force-encrypt-auth = true | false;
```

set force-flush-all Command

This command has the following format:

```
set force-flush-all = <bool>;
```

set group Command—Define an Access Control Group

Use the *set group* command to define an access control group.

This command is only useful for access control groups and using access control groups is deprecated. Use static or dynamic groups instead.

This command has the following format:

```
set group = {
    name = "group-name"
    users = DN-list
};
```

name = *group-name*

Specifies the name of this group.

users = *DN-list*

Lists the members of the group. This is a comma-separated list of distinguished names.

set history Command—Set the Number of Console Commands the history Command Will Display

The *set history* command determines the number of previously entered console commands the system will store and display when you enter the *history* command.

This command has the following format:

```
set history = number-of-commands;
```

number-of-commands

Sets the number of commands the system will store and display. You can check what this is set to at any time using the *get history* command.

Default: By default this is set to 15.

More information:

history Command—Display Previously Entered Console Commands (see page 45)
get history Command—Show How Many Console Commands the history Command Will Display (see page 39)

set hold-ldap-connections Command

The *set hold-ldap-connections* command controls whether a DSA clears the underlying TCP/IP connection after a bind refusal.

When set to *true*, this command prevents a DSA from clearing the underlying TCP/IP connection after a bind refusal.

This command has the following format:

```
set hold-ldap-connections = true | false;
```

set ignore-name-bindings Command—Allow the DSA to Operate Without Name Bindings

When set to *true*, this command lets the DXserver operate without name bindings.

This can be useful when the schema is imported from a directory that does not support name bindings.

This command has the following format:

```
set ignore-name-bindings = true | false;
```

set keep-order-of-values Command—Keep Order of Values for Attribute Types

The *set keep-order-of-values* command lets you specify attribute types whose values must be kept in the order in which they were added.

Note: This command will not work for operational attributes, distinguished values or object class values.

This command has the following format:

```
set keep-order-of-values = attr-list;
```

attr-list

A comma-separated list of the attribute types whose values must be kept in the order in which they were added.

Alternatively, specify *"none"* if you do not want to keep the order in which values are added for any attribute type.

set limit-search-exceptions Command—Let Some Users Bypass Complex Search Limits

The *set limit-search-exceptions* command lets some users bypass the limit on the number of complex searches or searches with no filter.

This is deprecated. Use search profiles instead.

This command has the following format:

```
set limit-search-exceptions = DN-list;
```

DN-list

Specifies the users and roles who can perform complex searches without limitations. This is a comma-separated list of distinguished names of users and roles.

set limit-search-exceptions-browse Command—Let Some Users Bypass Browse Search Limits

The *set limit-search-exceptions-browse* command lets all users browse a DSA that has a limit set.

This is deprecated. Use search profiles instead.

This command has the following format:

```
set limit-search-exceptions-browse = true|false;
```

true

Allows all users to browse the DSA when *limit-search* is in effect. General searching is still limited.

false

Prevents users from browsing the DSA when *limit-search* is in effect.

set log Command

The *set log* command opens a log file. Output will now be sent to it.

Note: The alarm log is always open.

This command has the following format:

```
set log-type-list;
```

log-type-list

A comma-separated list of one or more log files to be opened:

- alert-log
- cert-log
- connect-log
- diag-log
- query-log
- snmp-log
- stats-log
- summary-log
- time-log
- trace-log
- update-log
- warn-log

Note: For more information on the types of log you can set up, see the *Administration Guide*.

set lookup-cache Command—Enable Memory-Mapped File

This command enables the use of a memory-mapped file.

It must appear after the other cache settings. This is because the DSA loads the file as soon as it reads this command.

This command has the following format:

```
set lookup-cache = true | false;
```

true

Enables the memory-mapped file.

false

Unloads the file and disable the DSA

set max-bind-time Command

The *set max-bind-time* command specifies the maximum time a bind is held before being disconnected.

This command has the following format:

```
set max-bind-time = bind-time | none;
```

bind-time

Specifies the maximum time, in seconds, that any particular binding can last (a value of 0 or none means unlimited)

set max-cache-index-size Command

The *set max-cache-index-size* command sets the maximum number of entries (or distinct attribute values for each attribute type) that a cache DSA can contain.

This command has the following format:

```
set max-cache-index-size = number-entries;
```

number-entries

Specifies the maximum number of entries that can be present in an index in a cache DSA

Default: 8000000

set max-local-ops Command

This command sets the maximum number of concurrent operations the DSA processes for all users.

This command has the following format:

```
set max-local-ops = number-operations;
```

set max-op-size Command

This command sets the maximum number of entries that a search or list can return.

This command has the following format:

```
set max-op-size = number-entries | none;
```

number-entries

Specifies the maximum number of entries that any operation may return

none

Allows operations to continue with no size limitation

set max-op-time Command

This command sets the maximum time that any particular operation can last.

If the user has requested a time limit, and the *max-op-time* has also been set, the shorter setting takes effect.

This command has the following format:

```
set max-op-time = time | none;
```

time

Specifies the time in seconds for which any operation may last

none

Allows operations to continue with no time limitation

set max-pdu-size Command

The *set max-pdu-size* command sets the largest size that a protocol data unit may be to be accepted by a DSA. The default value is 0, meaning unlimited.

If this value has been set, you can retrieve it with the *get user* command.

This command has the following format:

```
set max-pdu-size = pdu-size;
```

pdu-size

Specifies the largest size (in bytes) that a protocol data unit may be to be accepted by a DSA

set max-users Command

The *set max-users* command specifies the maximum number of concurrent bindings, which is equivalent to the number of users that can concurrently bind to a DSA.

This command has the following format:

```
set max-users = number-bindings| none;
```

number-bindings

Specifies the maximum number of bindings.

none

Prevents any bindings to this DSA

Note: If you set max-users to 0 (zero) this has the effect of setting the maximum number of connections to about 4096. To prevent new connections to a DSA, use the set allow-binds (see page 54) command.

set mimic-netscape-for-siteminder Command

The *set mimic-netscape-for-siteminder* command makes CA Directory imitate a Netscape, iPlanet, or SunOne server.

This is necessary when you use a CA Directory DSA with SiteMinder, because SiteMinder uses a single thread for authentication. If SiteMinder does not detect a Netscape, iPlanet, or SunOne server, this thread serializes authentication binds to the directory, which slows performance.

To let SiteMinder process authentications asynchronously, the DSA must do both of the following:

- Pretend to be a Netscape server to Siteminder
- Process concurrent binds (see set concurrent-bind-user Command (see page 63))

This command has the following format:

```
set mimic-netscape-for-siteminder = true | false;
```

true

Sets the CA Directory DSA to imitate a Netscape directory for the purposes of SiteMinder authentication

false

Turns off this Netscape mimicry

set modify-on-add Command

When the *set modify-on-add* command is *true*, a *modifyTime* attribute is created when a new entry is created.

This command has the following format:

```
set modify-on-add = true | false;
```

set multi-write-dsp-idle-time Command

The set multi-write-dsp-idle-time command sets the idle time on links between multiwrite peer DSAs. If you do not use this command, the value in the set dsp-idle-time command is used instead.

This command is useful if the DSP idle time is set to a very low value. This can cause the link between a multiwrite DSA and its recovering peer DSA to time out before the DSA is recovered.

This command has the following syntax:

```
set multi-write-dsp-idle-time = time;
```

time

Specifies the idle time (in seconds) for links between multiwrite DSAs.

set multi-write-error-trap Command

The *set multi-write-error-trap* command determines whether an SNMP trap is sent when a replicated update is refused.

The trap will contain diagnostic information on why the update failed.

The SNMP trap is only sent if traps are configured.

When *multi-write-error-trap* is set to *true*, then the following rules apply:

- If a DSA cannot update a peer DSA because the peer is unreachable then the update request is queued and sent later: no trap is sent.
- If a DSA can send a request to the peer DSA (the peer is reachable), but the peer refuses the update, then the sending DSA sends a trap, because a refusal to accept a request shows some data inconsistency, which could be a serious issue.

This command has the following format:

```
set multi-write-error-trap = true | false;
```

set multi-write-queue Command

The *set multi-write-queue* command specifies the maximum number of multiwrite operations that will be queued when a peer DSA cannot be reached.

For more information, see Set Up Replication in the *Administration Guide*

This command has the following format:

```
set multi-write-queue = number-operations;
```

set multi-write-retry-time Command

The *set multi-write-retry-time* command defines the time period (in seconds) at which DXserver will attempt to bind to a Multiwrite peer which cannot be contacted. The default value is 60 seconds.

This command has the following format:

```
set multi-write-retry-time = retry-time;
```

set name-binding Command

Use the *set name-binding* command to define a name binding between two previously defined object classes. This defines the hierarchy of the directory.

A name binding consists of a name for the name binding, an object and its allowable parent, and an attribute that names the object

This command has the following format:

```
set name-binding OID = {
        name = binding-name
        parent-OC allowable-parent child-OC
        named-by attribute-list [ optional ]
};
```

OID

Specifies the object identifier of the attribute

name

The name of the name binding. This is often descriptive. It can be constructed by concatenating the names of the object classes being connected.

allowable-parent

Specifies the parent and child object classes

named-by

Lists the attributes that name an object of the child object class. Often only one attribute is listed.

optional

(Optional) Lists attributes that may optionally be appended to the list specified in named-by.

Example: Define Name Binding

In this example, a new definition (arbitrarily named *org-country*) states that you can place an organization object under a *country* object and that you must name it by the *organizationName* attribute. The definition *org-top* states that you can also place an organization object under a *top* object (that is, the root of the namespace) named by the *organizationName* attribute.

Multiple attributes can name an object, in which case you separate the attributes by commas. Additional naming attributes are optional when the keyword *optional* precedes them.

```
set name-binding x500nbind:2 = {
    name = org-top
    organization allowable-parent top
    named-by organizationName
};

set name-binding x500nbind:3 = {
    name = org-country
    organization allowable-parent country
    named-by organizationName
};
```

Example: Define Advanced Name Binding

```
set name-binding x500nbind:22 = {
name = orgUnit-orgPerson
organization allowable-parent organizationalUnit
named-by commonName optional surname
};
```

set object-class Command

The *set object-class* command defines an object class.

An object class is a fundamental part of the schema, and essential to the directory.

This command has the following format:

```
set object-class OID = {
    name        = DN
    ldap-names  = DN
    subclass-of DN
    kind        = structural | auxiliary | abstract
    must-contain attribute-list
    may-contain attribute-list
    description = "description"
```

name

The name of the object class. This is its formal name, and is often descriptive.

ldap-names

Alternative names for the object class. Think of these as nicknames-they can be used anywhere the name can be used. Often these are shorter.

subclass-of

Specifies the object class, or object classes, from which this object class inherits.

kind

Specifies the kind of object class.

must-contain *attribute-list*

Lists the attributes that must be supplied for every instance of this object class.

may-contain *attribute-list*

Lists the attributes that may be supplied in an instance of this object class.

description

A description of the object class.

Examples: set object-class Command

```
set object-class x500oc:5 = {
    name=       organizationalUnit
    subclass-of top
    kind=       structural
    must-contain organizationalUnitName
    may-contain organizationalAttributeSet
    description =       "X.500 Organizational Unit Object Class"

set object-class caOclass:0 = {
    name =      caOrgPerson
    subclass-of organizationalPerson
    kind =      structural
    may-contain caAttributeSet
    description =       "CA Organizational Person Object Class" };
```

set oid-prefix Command

The set oid-prefix defines an object identifier (OID) prefix.

An OID prefix consists of a name used to represent the portion of the object identifier common to multiple schema definition statements.

This command has the following format:

```
set oid-prefix sSetPrefix;
```

set op-attrs Command

The set-op-attrs command controls the creation and modification of operational attributes. It has the following format:

```
set op-attrs = { true | false } ;
```

true

The DSA automatically adds and updates operational attributes.

This is the default setting.

Note: If the *multi-write-disp-recovery* flag is set, the op-attrs parameter must be true.

false

Prevents the DSA from automatically creating operational attributes. However, some operational attributes will be created by the DSA if either DISP or multiwrite DISP recovery is configured.

set op-error-trap Command

The *set op-error-trap* command sets up the mechanism to hook into the handling of errors using SNMP traps. When set to true, this command turns on raising an operation error trap whenever and operation error occurs.

This command has the following format:

```
set op-error-trap = true | false;
```

set password-age Command

The *set password-age* command sets the number of days a password is valid.

If the password is not changed within the time set, the user account will expire. If this happens, an administrator must reset the password.

CA Directory uses the operational attribute *dxPwdLastChange* to maintain the password age.

This command has the following format:

```
set password-age = number-days | 0;
```

number-days

Specifies the number of days a password is valid.

0

(Default) Makes the password valid for an unlimited time.

set password-age-warning-period Command

The *set password-age-warning-period* command sets the number of days for which warnings about the password expiring are added to bind and compare responses.

For example, if a password has a life of 31 days, and you set the warning period to 5 days, the warning messages will be included in the last five days of the password's life.

Note: You can use this command only if the client is an LDAP client and it is aware of the Behera password policy request control.

This command has the following format:

```
set password-age-warning-period = number-days | 0;
```

number-days

Specifies the number of days during which a warning will appear.

0

(Default) Disables this feature.

set password-allow-ignore-expired Command

The *set password-allow-ignore-expired* command lets you bypass the expiration check of the password for some user accounts.

This command only takes effect for entries that include the attribute *dxPwdIgnoreExpired* and its value is *true*. This is useful for administrative accounts and accounts used by mission-critical applications.

This command has the following format:

```
set password-allow-ignore-expired = true | false;
```

true

Enables expiration bypassing. Administrators can now set an account to never expire by adding the attribute *dxPwdIgnoreExpired* and setting its value to *true*.

false

(Default) Disables expiration bypassing. All accounts expire as usual.

set password-allow-ignore-suspended Command

The *set password-allow-ignore-suspended* command lets you set an account to never be suspended.

Use this to protect an account from password lockout attacks. This is useful for administrative accounts and accounts used by mission-critical applications.

This command only takes effect for entries that include the attribute *dxPwdIgnoreSuspended* and its value is *true*.

This command has the following format:

```
set password-allow-ignore-suspended = true | false;
```

true

Enables suspension bypassing. Administrators can now set an account to never be suspended by adding the attribute *dxPwdIgnoreSuspended* and setting its value to *true*.

false

(Default) Disables suspension bypassing. All accounts can be suspended as usual.

set password-allow-locking Command

The *set password-allow-locking* command lets an administrator lock user accounts.

This command does not actually lock any accounts: this only takes effect for entries that include the attribute *dxPwdLocked*.

This command has the following format:

```
set password-allow-locking = true | false;
```

true

Enables password locking. Administrators can now lock accounts by including the attribute *dxPwdLocked*.

false

(Default) Disables password locking. No accounts can be manually locked.

set password-alpha Command

The *set password-alpha* command sets the minimum number of alphabetic characters in the password.

The alphabetic characters are:

- a-z
- A-Z

This command has the following format:

```
set password-alpha = number-chars | 0 ;
```

number-chars

Specifies the minimum number of alphabetic characters that a password must contain.

0

(Default) Disables this feature.

set password-alpha-num Command

The *set password-alpha-num* command sets the minimum number of alphanumeric characters in the password

The alphanumeric characters are:

- a-z
- A-Z
- 0-9

This command has the following format:

```
set password-alpha-num = number-chars | 0 ;
```

number-chars

Specifies the minimum number of alphanumeric characters that a password must contain.

0

(Default) Disables this feature.

set password-enforce-quality-on-reset Command

The *set password-enforce-quality-on-reset* command sets the DSA to enforce its password quality rules whenever any user changes any password.

If you do not set this, the password quality rules are not applied when an administrator changes a user's password.

This command has the following format:

```
set password-enforce-quality-on-reset = true | false;
```

true

Enforces the password quality checks on every password change.

false

(Default) Enforces the password quality checks only when a user changes their own password.

set password-force-change Command

The *set password-force-change* command forces users to change their passwords after their passwords have been reset.

Note: You can use this command only if the client is an LDAP client and it is aware of the Behera password policy request control.

When *set password-force-change* is set to *true* any bind by a new user or by a user with a reset password will be checked to see if it includes the Behera password policy control. This control is required so that the DSA can return the *password-force-change* control back to the client.

DAP binds do not support the Behera controls, which means that a user cannot bind to a DSA if *set password-force-change* is set to *true* and the password has been reset or the user's entry has just been created.

CA Directory uses the operational attribute *dxPwdMustChange* to force password changes.

This command has the following format:

```
set password-force-change = true | false;
```

true

Enables forced password changes. Users are prompted to change their password when they log in using a password that an administrator has changed.

false

(Default) Disables forced password changes. Users can continue to use a password that was changed by an administrator.

set password-grace-logins Command

The *set password-grace-logins* command sets the maximum number of times that the user can log in with their password after it has expired.

If the client is an LDAP client and the bind contains the Behera password-policy control, then if the password contained in the bind has expired, the bind-confirm returns an LDAP control containing the number of grace logins remaining.

If the client is not aware of the Behera password policy request control, grace logins will work, but the client will not be able to track how many grace logins are left.

CA Directory uses the operational attributes *dxPwdGraceLogins* and *dxPwdGraceUseTime* to maintain the grace login history.

This command has the following format:

```
set password-grace-logins = number-logins | 0 ;
```

number-logins

Specifies the number of times a user can log in with an expired password.

0

(Default) Disables this feature.

set password-history Command

The *set password-history* command sets the maximum number of entries to retain in the history. This prevents the user from re-using these passwords.

CA Directory uses the operational attribute *dxPwdHistory* to maintain the password history.

This command has the following format:

```
set password-history = number-passwords | 0;
```

number-passwords

Specifies the maximum number of passwords to retain in the history.

0

(Default) Disables this feature.

set password-last-use Command

The *set password-last-use* command sets the number of days a password remains valid if it is not used. If the password is not used for longer than the period you set, the user account is suspended.

CA Directory uses the operational attribute *dxPwdLoginTime* to record the last time the password was used.

This command has the following format:

```
set password-last-use = number-days | 0;
```

number-days

Specifies the number of days during which a password is not used but still remains valid.

0

(Default) Disables this feature.

set password-lowercase Command

The *set password-lowercase* command sets the minimum number of lowercase characters in the password.

The lowercase characters are *a-z*.

This command has the following format:

```
set password-lowercase = number-chars | 0 ;
```

number-chars

Specifies the minimum number of lowercase characters that a password must contain.

0

(Default) Disables this feature.

set password-max-length Command

The *set password-max-length* command sets the maximum length of a new password.

This command has the following format:

```
set password-max-length = number-chars | 0;
```

number-chars

Specifies the maximum number of characters a password may contain.

0

(Default) Disables this feature.

set password-max-repetition Command

The *set password-max-repetition* command sets the maximum number of repeated characters that a password may contain.

This command has the following format:

```
set password-max-repetition = number-chars | 0 ;
```

number-chars

Specifies the maximum number of repeated characters that a password may contain.

0

(Default) Disables this feature.

set password-max-substring-repetition Command

The *set password-max-substring-repetition* command sets the number of times that a substring can be repeated in new passwords.

The length of the substrings that will be checked for is set with the *set password-min-length-repeated-substring* command (see page 104).

This command has the following format:

```
set password-max-substring-repetition = number-repetitions;
```

number-repetitions

Specifies the number of times that substrings can be repeated in a password.

set password-max-suspension Command

The *set password-max-suspension* command sets the time after which a suspended password reactivates.

This setting only applies to accounts that were suspended because the user tried to log in too many times with the wrong credentials, as set with the *set password-retries* command.

CA Directory uses the operational attribute *dxPwdFailedTime* to record the time since the account was suspended due to failed login attempts.

This command has the following format:

```
set password-max-suspension = number-seconds | 0 ;
```

number-seconds

Specifies the time (in seconds) for which a suspended password remains suspended. After the time has passed, the account in active.

0

(Default) Disables this feature.

set password-mimic-netscape-response-controls Command

The *set password-mimic-netscape-response-controls* command adds LDAP response controls about password expiry to bind and compare responses. This mimics the way that Netscape directories work with LDAP password response controls.

For more information about the controls that are added to the responses, see the Sun Java System LDAP SDK for C Programming Guide.

This command has the following format:

```
set password-mimic-netscape-response-controls = true | false;
```

true

Includes the following LDAP response controls in bind and compare responses:

The control with the OID 2.16.840.1.113730.3.4.4

(Or LDAP_CONTROL_PWEXPIRED, as defined in the ldap.h header file)

This control is added to bind and confirm responses after the password has expired.

The control with the OID 2.16.840.1.113730.3.4.5

(Or LDAP_CONTROL_PWEXPIRING, as defined in the ldap.h header file)

The value for this control is the number of seconds before the password expires. This value is supplied in the *set password-age-warning-period* Command (see page 93).

false

(Default) No extra controls are added to bind and compare responses.

More information:

set password-age-warning-period Command (see page 93)

set password-min-age Command

The *set password-min-age* command sets a lockout period for changing the password. This is the time since a password was changed last before it can be changed again.

CA Directory uses the operational attribute *dxPwdLastChange* to maintain the lockout period.

This command has the following format:

```
set password-min-age = number-days | 0 ;
```

number-days

Specifies the time (in days) that must pass after the password is changed before it can be changed again.

0

(Default) Disables this feature.

set password-min-length Command

The *set password-min-length* command sets the minimum length of a new password.

This command has the following format:

```
set password-min-length = number-chars;
```

number-chars

Specifies the minimum password length.

Default: 6.

set password-min-length-repeated-substring Command

The *set password-min-length-repeated-substring* command sets the minimum length of substrings that will be checked.

The number of repetitions is set with the *set password-max-substring-repetition* command (see page 100).

If *set password-max-substring-repetition* is set to *true*, you *must* use the *set password-min-length-repeated-substring* command to set the substring length.

This command has the following format:

```
set password-min-length-repeated-substring = length-substring | 0;
```

length-substring

Specifies the minimum length of substrings that will be checked.

If *set password-max-substring-repetition* is set to *true* and you do not use this command to set the substring length, this value

0

Disables this feature.

set password-netscape-op-attrs Command

The *set password-netscape-op-attrs* command is useful if your CA Directory DSA is replicated to a third-party LDAP directory.

This command only affects the attributes that include the line *ldap-names = attribute-name* in their attribute definition in dxserver.dxc.

This command has the following format:

```
set password-netscape-op-attrs = true | false;
```

true

Replicates some CA Directory attributes to the other LDAP directory.

false

(Default)

set password-non-alpha Command

The *set password-non-alpha* command sets the minimum number of non-alphabetic characters a password may contain.

The non-alphabetic characters are:

- *space*
- ` ~ ! @ # $ % ^ & * ( ) - _ = + \ | [ ] { } ' " ; : / ? , . < >
- 0 1 2 3 4 5 6 7 8 9

This command has the following format:

```
set password-non-alpha = number-chars | 0;
```

number-chars

Specifies the minimum number of non-alphabetic characters that each password must contain.

0

(Default) Disables this feature.

set password-non-alpha-num Command

The *set password-non-alpha-numeric* command sets the minimum number of non-alphanumeric characters a password may contain.

The non-alphanumeric characters are:

- *space*
- ` ~ ! @ # $ % ^ & * ( ) - _ = + \ | [ ] { } ' " ; : / ? , . < >

This command has the following format:

```
set password-non-alpha-num = number-chars | 0;
```

number-chars

Specifies the minimum number of non-alphanumeric characters that each password must contain.

0

(Default) Disables this feature.

set password-numeric Command

The *set password-numeric* command sets the minimum number of numeric characters a password must contain.

The numeric characters are 0-9.

This command has the following format:

```
set password-numeric = number-chars | 0;
```

number-chars

Specifies the minimum number of numeric characters that each password must contain.

0

(Default) Disables this feature.

set password-policy Command

The *set password-policy* command specifies whether password management is enabled.

This command has the following format:

```
set password-policy = true | false;
```

true

Enables password policies. Any password rules you have set are active.

false

(Default) Disables password policies. No password rules are active.

set password-proxy-user Command

The *set password-proxy-user* command specifies the distinguished name of the password proxy user. This proxy user is a user account that performs password comparisons and updates on behalf of another user.

When an application binds as this password proxy user, the password policy will be applied to password compares and modifications.

This can be useful if an application that uses the directory can only have one connection to the directory.

This command has the following format:

```
set password-proxy-user = DN;
```

DN

Specifies the DN of the password proxy user.

set password-retries Command

The *set password-retries* command sets the number of consecutive failed login attempts before the user account is suspended.

CA Directory uses the operational attribute *dxPwdFailedAttempts* to record the number of failed login attempts.

This command has the following format:

```
set password-retries = number-retries;
```

number-retries

Specifies the number of times a user may attempt to log in with an incorrect password before their account is suspended.

Default: 3

set password-storage Command

The *set password-storage* command lets you select a hashing method for passwords stored in the directory.

This command has the following format:

```
set password-storage = sha-1 | ssha-1 | sha-512 | ssha-512 | crypt | md5 | smd5 | none;
```

sha-1

(Default) Hashes the password using the SHA-1 algorithm

ssha-1

Hashes the password using the Salted SHA-1 algorithm. This algorithm produces a different hash even for the same clear text password, which is more secure.

sha-512

Hashes the password using the SHA-512 algorithm

ssha-512

Hashes the password using the Salted SHA-512 algorithm

crypt

Hashes the password using the UNIX crypt method.

md5

Hashes the password using the Message Digest algorithm

smd5

Hashes the password using the Salted Message Digest algorithm

none

Passwords are not hashed. This should only be used for testing.

set password-substring-attrs Command

The *set password-substring-attrs* command defines whether the password can contain the values of attributes in the user's entry.

```
set password-substring-attrs = attribute-list;
```

attribute-list

Specifies the attributes that the password will be checked against. This is a comma-separated list of attributes.

set password-suspended-trap Command

The *set password-suspended-trap* command raises an SNMP trap when a password is suspended.

This command has the following format:

```
set password-suspended-trap = true | false;
```

true

Raises an SNMP trap when a password is suspended.

false

(Default) Disables the SNMP trap.

set password-uppercase Command

The *set password-uppercase* command sets the minimum number of uppercase characters in the password.

The uppercase characters are *A-Z*.

This command has the following format:

```
set password-uppercase = number-chars | 0 ;
```

number-chars

Specifies the minimum number of uppercase characters that a password must contain.

0

(Default) Disables this feature.

set password-username-substring Command

The *set password-username-substring* command defines whether the password can contain the user's name.

When this is set to true, the password must not be a substring of the last RDN in the user's DN.

This command has the following format:

```
set password-username-substring = true | false;
```

true

Checks all new passwords against the user's name, and rejects any password that is a substring of the user's name.

false

(Default) Disables the username check.

set persistent-search Command

The *set persistent-search* command allows CA Directory to use the LDAP persistent search control. Persistent searches are defined in the LDAP V3 extension

This is useful if an application opens persistent searches so that it is notified as and when operations happen. For more information about LDAP persistent searches, see Persistent Search.

This command has the following format:

```
set persistent-search = true | false;
```

true

Allows CA Directory to respond to searches that use the LDAP V3 persistent search control

false

Disables persistent searching

set protected-items Command—Configure Protected Items Access Level Controls

The *set protected-items* command lets you protect specific subtrees, entries, or particular attributes in a subtree or entry. Use protected-items controls to protect some attributes in a part of the DIT.

This command denies (takes away) specified access rights that have been granted at the registered users and public users access level

Access rights denied at this access level can be granted by rules at the administrative users and super users access levels.

Access control rules are effective only if you enable access controls.

This command has the following format:

```
set protected-items [tag] = {
    [users]
    [scope]
    [attrs      = attribute-list]
    [perms      = permission-list]
    [validity   = [start hhmm end hhmm] [on day]]
};
```

tag

(Optional) Defines a name for this rule.

users

(Optional) Identifies the users that are to be denied access to the item. Use one of the following:

own-entry

Specifies that this rule restricts users' access to their own entry.

own-subtree

Specifies that this rule restricts users' access to the subtree beneath their own entry.

user = *DN*

Specifies the user whose access this rule restricts.

role = *DN*

Specifies the role whose access this rule restricts.

group = *group-name*

Specifies the access control group whose access this rule restricts.

user-subtree = *DN*

Specifies the subtree of users whose access this rule restricts.

scope

(Optional) Defines the part of the DIT that this rule applies to.

Use one of the following:

entry = *DN*

Specifies the entry that this rule protects.

If entry is specified and is not a leaf node then the user still has the right to use the entry in a path to a lower level entry, so normally you only specify *entry* if it is a leaf node.

subtree = *DN*

Specifies the subtree that this rule protects.

attrs = *attribute-list*

(Optional) Defines the attributes or attribute set to which this rule applies, where *attribute-list* is a comma-separated list of attribute names.

If *attrs* is not specified, then the access rule applies to the whole entry.

perms = *permission-list*

If *perms* is not specified then the command will deny all permissions.

Normally you will not specify *perms*.

all

Denies users all permissions over the scope.

read

Denies users read permission over the scope. This has the same effect as specifying *perms=all.*

add

Denies users permission to add to the information defined in the scope.

remove

Denies users permission to delete entries defined in the scope.

modify

Denies users permission to change information defined in the scope.

rename

Denies users permission to rename the entries defined in the scope.

validity = [start *hhmm* end *hhmm*] [on *day*]

(Optional) Defines the period during which this rule is valid. Use any of the following:

start *hhmm* end *hhmm*

Defines the start and end of the period during which this rule is valid.

on *day*

Defines the day on which this rule is valid, where *day* is a string like 12345 or 67 (1 is Monday).

Example: Protect a Subtree

The command in this example makes a subtree invisible:

```
set protected-items hide-finance-from-employees = {
 group   = employees"
 subtree= <c AU"><o Democorp"><ou Finance">
};
```

Example: Protect an Entry

The command in this example could be used to hide some management information about a DSA definition stored within the directory:

```
set protected-items hide-schema-from-employees" = {
 role    = employees"
 entry = <c AU"><o Democorp"><ou Schema">
};
```

The specified entry is invisible to members of the employees role (unless a higher precedence access control rule grants them some access).

Example: Protect Some Attributes

The command in this example protects the *homePhone* and *userPassword* attributes in any entries in the Democorp subtree that have these attributes.

These attributes are visible to any super users and administrative users that have the Democorp subtree in their scope, but the attributes are hidden from all other users:

```
set protected-items hide-passwords-and-home-phone" = {
 subtree= <c AU"><o Democorp">
 attrs   = homePhone, userPassword
};
```

Example: Let Users View a Whole Entry and Modify Some Attributes

This example shows how to give users update access to most attributes within an entry but prevent the update of a small number of attributes.

This problem is made more complex because the list of attributes that can be updated can grow, for example, as more attributes are added to the entry.

- Providing *reg-user* access to the entry will give read-only access.
- Providing *admin-user* access will give update access.
- Setting *protected-items* on some attributes will prevent updates, but not by users with *admin-user* rights. It will also prevent reads by users with *public-user* or *reg-user* rights.

A solution to this problem is to use the optional permissions in the access control rules. The following access rule will let all users in the subtree modify all attributes in their own entry:

```
set reg-user = {
 own-entry
 subtree= <o test>
 perms  = modify
};
```

To prevent update access to some of these attributes, use the following command:

```
set protected-items = {
 own-entry
 subtree= <o test>
 attrs  = attr1, attr2, ...
 perms  = modify
};
```

This prevents the user modifying the attributes listed, but it does not prevent read access. This is because the only permission denied is *modify*.

Example: Let Users Modify All Attributes in Their Own Entry Except "role"

In this example, the *set reg-user* rule gives users modification rights to all attributes in their own entry, and the *protected-items* rule takes away modification rights for just the role attribute. The result is that users can modify all attributes in their own entries except "role", which they can read:

```
set reg-user = {
 own-entry
 subtree= <o Democorp>
 perms  = modify
};

set protected-items = {
 own-entry
 subtree= <o DemoCorp>
 attrs  = role
 perms  = modify
};
```

Without the *perms = modify* in the *set protected-items* rule, the user would be denied all access to the *role* attribute (including read access).

set prune-oc-parents Command

This command has the following format:

```
set prune-oc-parents = true | false;
```

set public-user Command—Configure Anonymous User Access Level Rights

A public user is a user who is anonymous, so the set of "Public users" consists of all users who have not been authenticated.

This command grants specified access rights at the public user access level, to all users, over a specified scope.

Any access that is granted by this command applies to public users, and by extension to all users. That is, a user who is authenticated can do anything that a public user can do.

Access rights granted at this access level can be taken away by access control rules defined at the protected items access level.

Access control rules are effective only if you enable access controls. If access controls are not enabled, then public users have full permission over the whole directory.

This command has the following format:

```
set public-user [tag] = {
    scope
    [attrs      = attribute-list]
    [perms      = permission-list]
    [validity   = [start hhmm end hhmm] [on day]]
};
```

tag

(Optional) Defines a name for this rule.

scope

Defines the area of the DIT that this rule gives access to, where *scope* is one of the following:

entry = *DN*

Specifies the entry that this rule grants access to.

subtree = *DN*

Specifies the subtree that this rule grants access to.

attrs = *attribute-list*

(Optional) Defines the attributes or attribute set to which this rule applies, where *attribute-list* is a comma-separated list of attribute names.

If *attrs* is not specified, then the access rule applies to the whole entry. *add* and *remove* permissions require that *attrs* is not specified.

perms = *permission-list*

(Optional) Specifies the permissions (access rights) that this rule grants to public users for the *scope*.

If *perms* is not specified, then read access permission is granted.

permission-list is a comma-separated list of one or more of the following:

all

Specifies that public users have all available permissions over the scope. This option implies all of the permissions listed below.

read

Specifies that public users can read the information defined in the scope.

add

Specifies that public users can add to the information defined in the scope. This also grants read permission.

remove

Specifies that public users can delete entries defined in the scope. This also grants read permission.

modify

Specifies that public users can change information defined in the scope. This also grants read permission.

rename

Specifies that public users can rename the entries defined in the scope. This also grants read permission.

validity = [start *hhmm* end *hhmm*] [on *day*]

(Optional) Defines the period during which this rule is valid. Use any of the following:

start *hhmm* end *hhmm*

Defines the start and end of the period during which this rule is valid.

on *day*

Defines the day on which this rule is valid, where *day* is a string like 12345 or 67 (1 is Monday).

Example: Let Anonymous Users Read Attributes in a Subtree

In the following example, all users can view the name, telephone number, and X.400 mail addresses in the Phone List subtree:

```
set public-user public-attr" = {
 subtree= <c AU"><o Democorp"><ou Phone List">
 attrs  = telephoneNumber, commonName, surname, mhsORAddresses
};
```

Example: Give Public-User Privileges to Members of a Role

In the following example, all users in the role *cell-research* have read privileges on the directory R&D subtree:

```
set public-user cell-research" = {
 role = <c AU"><o Democorp"><ou roles"><cn cell-research">
 subtree = <c AU"><o Democorp"><ou R&D">
};
```

set query-log-show-eis Command—Show or Hide eis Information in Query log

The *set query-log-show-eis* command shows or hides eis data in the query log.

If *query-log-show-eis* is set to *true*, then query logs include the attribute names contained in a search requests entry information selection. The default behavior is to show only a count of attributes returned.

This command has the following format:

```
set query-log-show-eis = true | false;
```

set referential-integrity Command

Use the *set referential-integrity* command to define a referential integrity rule. A referential integrity rule is useful if you want to ensure that when you delete an entry, references to that entry are also deleted.

This command has the following format:

```
set referential-integrity rulename = {
    subtree = subtreeDN
    reference-subtree = referenceDN
    direct-attr = memberAttribute | indirect-attr = entryAttribute reference-attr =
    referenceAttribute
    };
```

rulename

Defines the name of the integrity rule.

subtreeDN

Specifies the subtree that contains the entries whose removal triggers this rule. When an entry in this subtree is deleted the DSA runs this integrity rule.

referenceDN

Specifies the subtree to be searched when the DSA runs this integrity rule.

memberAttribute

Specifies an attribute that may exist in one or more entries in the *referenceDN* subtree. The DSA finds all attributes named *memberAttribute* that are in the *referenceDN* subtree and have DN syntax. For each of these attributes, the DSA removes the value if it equals the DN of the entry that was deleted.

entryAttribute

Specifies an attribute name in the deleted entry. When the DSA deletes an entry, it retrieves the value of that entry's *entryAttribute.*

referenceAttribute

Specifies the attribute name that the DSA uses to search for references. The DSA finds all attributes named *referenceAttribute* that are in the *referenceDN* subtree. For each of these attributes, the DSA removes the value if it equals the value of *entryAttribute*.

Example: Define Direct Referential Integrity

The following referential-integrity rule is defined:

```
set referential-integrity groupsRule ={
    subtree=<c AU><o Users>
    reference-subtree =<c AU><o Groups>
    direct-attr = member
    };
```

The DSA receives a request to delete an entry *cn=Craig Link,o=Users,c=AU.*

After it has deleted the entry, the DSA deletes the value *cn=Craig Link,o=Users,c=AU* from all attributes that satisfy all the following conditions:

- Are in the subtree *o=Groups,c=AU*
- Are named *member*
- Have a DN syntax
- Have a value *cn=Craig Link,o=Users,c=AU*

Example: Define Indirect Referential Integrity

The following referential-integrity rule is defined:

```
set referential-integrity groupsRule ={
    subtree=<c AU><o Users>
    reference-subtree =<c AU><o Groups>
    indirect-attr = userID
    reference-attr=guid
    };
```

The DSA receives a request to delete an entry *cn=Craig Link,o=Users,c=AU.*

After it has deleted the entry, the DSA deletes the value of the *userID* attribute in the deleted entry from all attributes that satisfy all the following conditions:

- Are in the subtree *o=Groups,c=AU*
- Are named *guid*
- Have a value equal to the value of the *userID* attribute in the deleted entry

set rdn-order Command—Specify Attribute Order

The *set rdn-order* command lets you specify the order in which the attributes appear in the RDN, if you have a multi-valued RDN. You can list as many attributes as you want and separate them with a comma.

This command has the following syntax:

```
set rdn-order = attribute, attribute, attribute;
```

set reg-user Command—Configure Registered User Access Level Rights

This command grants specified access rights at the registered user access level, to specified users, over a specified scope.

Access rights granted at this access level can be taken away by access control rules defined at the protected items access level.

Access control rules are effective only if you enable access controls.

This command has the following format:

```
set reg-user [tag] = {
    users
    scope
    [attrs      = attribute-list]
    [auth-level = simple | ssl-auth]
    [perms      = permission-list]
    [validity   = [start hhmm end hhmm] [on day]]
};
```

tag

(Optional) Defines a name for this rule.

users

Defines the users that this rule applies to, where *users* is one of the following:

user = *DN*

Defines the user that this rule applies to.

role = *DN*

Defines the role that this rule applies to.

group = *group-name*

Defines the access control group that this rule applies to. Use of access control groups is deprecated, so use of this option is also deprecated.

user-subtree = *DN*

Defines the top of the subtree of users that this rule applies to.

own-entry

Specifies that the users defined in *scope* have access to their own entries only.

own-subtree

Specifies that the users defined in *scope* have access to their own entries and any entries below their own entry.

scope

Defines the area of the DIT that this rule gives access to, where *scope* is one of the following:

entry = *DN*

Specifies the entry that this rule grants access to.

subtree = *DN*

Specifies the subtree that this rule grants access to.

attrs = *attribute-list*

(Optional) Defines the attributes or attribute set to which this rule applies, where *attribute-list* is a comma-separated list of attribute names.

If *attrs* is not specified, then the access rule applies to the whole entry. *add* and *remove* permissions require that *attrs* is not specified.

perms = *permission-list*

(Optional) Specifies the permissions (access rights) that this rule grants to the *users* for the *scope*.

If *perms* is not specified, then read access permission is granted.

permission-list is a comma-separated list of one or more of the following:

all

Grants users all permissions over the scope.

read

Grants users permission to read the information defined in the scope.

add

Grants users permission to add to the information defined in the scope. This also grants read permission.

remove

Grants users permission to delete entries defined in the scope. This also grants read permission.

modify

Grants users permission to change information defined in the scope. This also grants read permission.

rename

Grants users permission to rename the entries defined in the scope. This also grants read permission.

auth-level = simple | ssl-auth

(Optional) Specifies the level of authentication required. If you use this option, use one of the following:

simple

Specifies that this rule only applies to users who bind using simple authentication (username and password).

ssl-auth

Specifies that this rule only applies to users who bind using SSL authentication.

validity = [start *hhmm* end *hhmm*] [on *day*]

(Optional) Defines the period during which this rule is valid. Use any of the following:

start *hhmm* end *hhmm*

Defines the start and end of the period during which this rule is valid.

on *day*

Defines the day on which this rule is valid, where *day* is a string like 12345 or 67 (1 is Monday).

Example: Give Read Access to All Users in a Subtree

In the following example, all the users in the R&D subtree can read the Democorp subtree:

```
set reg-user R&D-Users"= {
 user-subtree = <c AU"><o Democorp"><ou R&D">
 subtree = <c AU"><o Democorp">
};
```

Example: Give Read Access to an Entry

In the following example, all users in the group staff have read privileges on the Democorp entry:

```
set reg-user democorp-staff" = {
 group = staff"
 entry = <c AU"><o Democorp">
};
```

Example: Let All Users Read Some Attributes in Their Own Entry

The following example lets any user in the subtree AU/Democorp view only the selected attributes in their entry:

```
set reg-user = {
    own-entry
    subtree = <c AU"><o Democorp">
    attrs = telephoneNumber, commonName, surname, title, mhsORAddresses, odEmail
};
```

Example: Let All Users Read and Modify Some Attributes

In this example, all Democorp users can browse all entries in the subtree Democorp; however, when they read or search for an entry in the subtree, only those attributes that you declare are visible.

The users also have modify privileges on the listed attributes for all entries in the subtree:

```
set reg-user self-view" = {
 user-subtree = <c AU><o Democorp>
 subtree = <c AU"><o Democorp">
 attrs = telephoneNumber, commonName, surname, title, mhsORAddresses, dcEmail
 perms = modify
};
```

set relaxed-not-search Command

The *set relaxed-not-search* command lets you interpret NOT in the non-standard manner supported by some other directories.

Normally, CA Directory handles NOT in LDAP searches according to this standard: ISO/IEC 9594-3 : 2001 (E) Section 7.8.2 Filter item, 4th paragraph. However, some other directories handle this differently. If you set this command to true, CA Directory will handle NOT in searches in this non-standard way.

For example, if *relaxed-not-search* is set to *true*, and you search for "description not equal M*", the search returns the entries that contain no description, and the entries that have a description and the description does not start with M.

If you run the same search while *relaxed-not-search* is not set or is set to *false*, the same search returns only those entries that have a description and the description does not start with M.

To display the value of this setting, use the *get oper* command.

This command has the following format:

```
set relaxed-not-search = true | false;
```

true

Interprets NOT in searches in the non-standard way of some other directories

false

Interprets NOT in searches as specified in ISO/IEC 9594-3 : 2001

set return-oc-parents Command

This command has the following format:

```
set return-oc-parents = <bool>;
```

set role-subtree Command

The *set role-subtree* command specifies the DN where the roles are defined.

Together with use-roles, it is used to set role-based access controls.

This command has the following format:

```
set role-subtree = DN;
```

DN

Specifies the distinguished name at the top of the subtree that stores the roles.

More information:

set use-roles Command (see page 139)

set ssl Command—Configure SSL

The *set ssl* command lets you configure the behaviour of SSL.

The command only takes effect when the dxserver starts. If you change SSL parameters using the DSA console, values are not changed and the following warning is logged to the warn file:

```
WARN : Cannot change SSL params once set
```

This command has the following format:

```
set ssl = { cert-dir = certificate_directory ca-file = certification_authority [cipher
= cipher] [protocol = tls] [fips = true] [pin = pin] [lib = library] [slot = slot]
} ;
```

cert-dir

Identifies the directory that contains certificate and private-key files in PEM format.

ca-file

Identifies the file that contains trusted certification authority certificates in PEM format.

cipher

(Optional) Specifies the ciphers that will be used for SSL and TLS connections.

protocol

(Optional) Instructs CA Directory to use TLS instead of SSL 3.0.

Limits: tls

Default: SSL 3.0

fips

(Optional) Specifies to run SSL in FIPS only mode. In this mode, the DSA will only accept FIPS compliant ciphers.

Limits: True

Default: False

pin

(Optional) Specifies the hardware security module (HSM) user PIN. If specified, the private key is used through the HSM. For example:

```
pin=1234
```

Limits: Valid PIN

lib

(Optional) Specifies the file containing the pks#11 library supplied by the HSM vendor. For example:

```
lib="C:\Program Files\Eracom\ProtectToolkit C Runtime\cryptoki.dll"
```

Limits: Valid path and dll file name

slot

(Optional) Specifies the slot location in the HSM where the corresponding private keys are stored. For example:

```
slot=2
```

Limits: Valid slot number

set ssl-auth-bypass-entry-check Command

The *set ssl-auth-bypass-entry-check* command allows a bind to skip server authentication..

When set to *true*, this command turns off automatic checking for the existence of an entry named by the subject held in the certificate.

This command has the following format:

```
set ssl-auth-bypass-entry-check = true | false;
```

set super-user Command—Configure Super User Access Level Rights

This command grants all access rights (permissions) at the super user access level, to specified users. The scope is a user's own entry, or own subtree, or the whole directory.

Access rights granted at this access level cannot be taken away by other access control rules.

Access control rules are effective only if you enable access controls.

This command has the following format:

```
set super-user [tag] = {
    users
    [auth-level  = simple | ssl-auth]
    [validity    = [start hhmm end hhmm] [on day]]
};
```

tag

(Optional) Defines a name for this rule.

users

Defines the users that this rule applies to, where *users* is one of the following:

user = *DN*

Defines the user that this rule applies to.

role = *DN*

Defines the role that this rule applies to.

group = *group-name*

Defines the access control group that this rule applies to. Use of access control groups is deprecated, so use of this option is also deprecated.

user-subtree = *DN*

Defines the top of the subtree of users that this rule applies to.

own-entry

Specifies that the users defined in *scope* have super user access to their own entries only.

own-subtree

Specifies that the users defined in *scope* have super user access to their own entries and any entries below their own entry.

auth-level = simple | ssl-auth

(Optional) Specifies the level of authentication required. If you use this option, use one of the following:

simple

Specifies that this rule only applies to users that bind using simple authentication (username and password).

ssl-auth

Specifies that this rule only applies to users that bind using SSL authentication.

validity = [start *hhmm* end *hhmm*] [on *day*]

(Optional) Defines the period during which this rule is valid. Use any of the following:

start *hhmm* end *hhmm*

Defines the start and end of the period during which this rule is valid.

on *day*

Defines the day on which this rule is valid, where *day* is a string like 12345 or 67 (1 is Monday).

Example: Give Super User Privileges to One User

The following command defines a single user with super user privileges:

```
set super-user dsa-manager" = {
 user = <c AU"><o Democorp"><commonName DSA manager">
};
```

Example: Give Users Super User Rights to Their Own Entry Only

The following command gives all users in the domain of this DSA super user privileges on their own entry from 0800 hours to 1800 hours on Monday (day 1) to Friday (day 5):

```
set super-user self" = { own-entry
 validity = ( start 0800 end 1800 on 12345 )
};
```

When you include this command in an access.dxc file that multiple DSAs source, all users in the domains of those DSAs will have super user privileges on their own entries.

The *own-entry* and *own-subtree* options are the only types of super user rule that do not grant the user access to all parts of the DSA.

set syntax-alias Command

Syntax aliasing is useful if you add a new schema object definition that uses an attribute syntax that is not supported by CA Directory.

This command has the following format:

```
set syntax-alias oid = { name = alias-name alias-for = syntax };
```

alias-for = *syntax*

Specifies one of the supported syntaxes.

Example: Use the *set syntax-alias* Command

```
set syntax-alias (1.3.6.1.4.1.1466.115.121.1.18) = {
  name = dlSubmitPermissions
  alias-for = caseIgnoreString
};
```

More information:

Supported Attribute Syntaxes (see page 259)

set time-log-search-threshold Command—Limit The Display of Compare and Search Operations in the Time Log

The *set time-log-search-threshold* command limits the display of compare and search operations in the time log output. Use this command to set the shortest duration that you want to see displayed.

To display the value of this setting, use the *get user* command.

This command has the following format:

```
set time-log-search-threshold = time | none ;
```

time

(In milliseconds) Only those compare and search operations with a duration greater than or equal to this value are displayed.

none

No compare or search operations will be listed in the time log.

set time-log-update-threshold Command—Limit the Display of Update Operations in the Time Log

The *set time-log-update-threshold* command limits the display of add, modify, moddn, and remove operations in the time log output. Use this command to set the shortest duration that you want to see displayed.

To display the value of this setting, use the *get user* command.

This command has the following format:

```
set time-log-update-threshold = time | none ;
```

time

(In milliseconds) Only those add, modify, moddn, and remove operations with a duration greater than or equal to this value are displayed.

none

No compare or search operations will be listed in the time log.

set trace Command—Define Trace Levels

The *set trace* command enables or disables tracing.

Trace options in the *set trace* command are not cumulative and override options set in previous *set trace* commands.

This command has the following format:

```
set trace = none | trace-level-list;
```

none

Turns off all tracing

trace-level-list

A comma-separated list of one or more of the following trace options:

alert

Displays authentication errors.

cert

Displays certificate operations.

connect

Displays connections.

diag

Traces local DSA operations that were refused.

dsa

Similar to the x500 trace, but also includes tracing of the module flow inside the DSA. Use this trace level when you are trying to identify a problem with the directory.

error

Displays error messages of very high severity.

These events that may impact on the ability of the DSA to perform a requested operation. These events are usually more serious than those reported under warn.

This is the default trace level. It has the lowest impact on performance, and we recommend that you use this level during normal operation.

ldap

Traces detailed LDAP operations. The output can become quite large when searches return a large number of entries.

limit

Traces any violation of size or time limits.

query

Displays a one-line summary containing the server request and result.

stack

Displays detailed protocol tracing. The output can become large.

stats

Displays statistical information for each minute the DSA is not idle.

summary

Displays a one-line summary containing the service request and result.

time

Displays the time taken for successful operations.

To send this output to a separate file, use the *set time-log* command.

update

Displays update operations-add, delete, modify, and rename.

warn

Displays error messages of moderately high severity.

Warning messages usually represent a user error, rather than a problem with the DSA. This used to be the default trace level, but the default is now *error*.

x500

Displays the full details of the service request, confirmation, or error. This traces DAP, DSP, and LDAP operations. The output can become large when searches return a large number of entries.

Example: Set Multiple Trace Levels

You can set more than one trace level. This means that the trace includes all of the information usually captured by each trace level.

The following command sets the DSA to capture any high-level error messages, plus any violations of time or size limits:

```
set trace = time, error;
```

Example: Set Non-Cumulative Tracing Levels

This example shows a set of tracing commands. Because the *set trace* command is not cumulative, only the last command affects the final tracing level. In the following example, the final trace option is *summary*:

```
set trace = time, error;

set trace = summary;
```

The *time* and *error* options are turned off by the second command.

set transparent-routing Command

The *set transparent-routing* command enables a router DSA to route LDAP queries without knowing the related schema.

This command has the following format:

```
set transparent-routing = <bool>;
```

set trap-on-update Command

The *set trap-on-update* command is a mechanism that can be used to hook into the processing of updates on the DSA using SNMP traps. When set to true, this command will raise an SNMP trap whenever an add, delete, modify, or rename event occurs for an entry. This means you can, for example, use the set trap-on-update command to log actions or trigger other events on a third-party system.

This command has the following format:

```
set trap-on-update = true | false;
```

set trap-on-update-verbose Command

The *set trap-on-update-verbose* command extends the *set trap-on-update-verbose* command.When set to true, this verbose command will raise an SNMP trap whenever an add, delete, modify, or rename event occurs and will also record information about what attribute was changed for an entry, and whether the request for that change originated on another DSA (assuming you have multi-write configured).

Note: You must have set trap-on-update Command set to true, before you can use this command.

This command has the following format:

```
set trap-on-update-verbose = true | false;
```

set trust-sasl-proxy Command

The *set trust-sasl-proxy* command specifies the distinguished name of the trusted proxy.

This command has the following format:

```
set trust-sasl-proxy = DN-list;
```

DN-list

Specifies the distinguished names of one or more trusted proxies in a comma-separated list.

set unique-attrs Command—Enable Checks for Uniqueness of Attribute Values

The *set unique-attrs* command enables uniqueness checking for new attribute values. These checks only apply to attribute values in the same DSA.

You can specify the subtree that each attribute must be unique within.

This command has the following format:

```
set unique-attrs = attribute [subtree = DN] [,attribute [subtree = DN]] [...] ;
```

attribute

Specifies an attribute that should have unique values.

subtree = *DN*

(Optional) Specifies the subtree that the attribute should be unique in.

If you don't specify this, the attribute will be unique in the subtree specified by the *set unique-attr-subtree* command. If you have not defined the *set unique-attr-subtree* command, the attribute will be unique within the local prefix.

Example: Make Two Attributes Unique Within a Specified Subtree

The following commands specify that the values of the *uid* and *cn* attributes must be unique within the Corporate subtree in Democorp:

```
set unique-attrs-subtree = <o democorp><ou corporate>;
set unique-attrs = uid, cn;
```

Example: Make Two Attributes Unique in Different Subtrees

In this example, the values of the *uid* and *phoneNumber* attributes must be unique within the Corporate subtree. However, the values of the *cn* attribute must be unique within the Staff subtree:

```
set unique-attrs-subtree = <o democorp><ou corporate>;
set unique-attrs = uid, phoneNumber, cn subtree = <o democorp><ou Sales>;
```

set unique-attrs-subtree Command—Enable Checks for Uniqueness Within a Subtree

If you have already set up a DSA to check whether attribute values are unique within the DSA, you can also set up checking within a subtree. All attribute values in all DSAs that serve that part of the namespace are now checked for uniqueness.

This command has the following format:

```
set unique-attrs-subtree = DN;
```

DN

Specifies the distinguished name of the base for unique value checking.

set update-log-show-values Command

The *set update-log-show-values* command lets you include attribute values in the update log.

The log will not include values for the following:

- Operational attributes
- DSA-specific attributes (such as password policy)
- The *userPassword* attribute
- Binary values

Note: The size of a line written to this log is restricted to 4096 bytes, so large values may be truncated.

This command has the following format:

```
set update-log-show-values = true | false;
```

true

Includes attribute values in the update log.

false

(Default) Does not include the attribute values.

set use-dynamic-roles Command

The *set use-dynamic-roles* command enables or disables dynamic roles.

To use dynamic roles, you need to set up dynamic groups, and then use the *set use-dynamic-roles* and *set role-subtree* commands.

This command has the following format:

```
set use-dynamic-roles = true | false;
```

More information:

set role-subtree Command (see page 127)

set use-roles Command

The *set use-roles* command enables or disables static roles.

To use static roles, you need to set up static groups, and then use the *set use-roles* and *set role-subtree* commands.

This command has the following format:

```
set use-roles = true | false;
```

More information:

set role-subtree Command (see page 127)

set user-idle-time Command

The *set user-idle-time* command specifies the maximum time a user is idle before being disconnected.

When a user is idle for too long, that user is disconnected. This reduces the number of users connected and lets new users connect to the DSA.

This command has the following format:

```
set user-idle-time = time;
```

time

Specifies the maximum idle time in seconds.

set user-threads Command—Define the Number of Threads for Requests

The *set user-threads* command defines the number of threads available on a DSA to user requests. If a request arrives and no thread is available the request is put in a queue. If a user thread is available, the DSA fulfills this request.

On a multi-CPU host, eight is a suitable initial setting. For a single CPU host, do not set user-threads, or set it to 1. The default is 8.

This command has the following syntax:

```
set user-threads = n;
```

n

Defines the number of user threads.

set view Command—Define a View

To define a view to the DSA, you use the *set view* command.

The syntax of the command is as follows:

```
set view viewName = {
    description="description"
    entry = ViewDN
    [options = [collapse-result | collapse-result-under-entry]
        [, remap-originator] [, view-entry-access-controls] ]

    (phase=1
        subtree = phaseDN
        [scope = {subtree | base | one-level}]
        [filter = phaseFilter]
        [eis = [prefix.]attributeName[.suffix]
        [,[prefix.]attributeName[.suffix]]...]
        [allow-attr = allowAttribute pllow-target = allowTarget]
        [prune-attr = pruneAttribute prune-target = pruneTarget]
        [options = [ignore-from-result] [result-required] [prune-from-result]]
        )

    [if (condition) |
        else if (condition) |
        else
        ]

    [,(phase=2
        subtree = phase_DN
        [scope = {subtree | base | one-level}]
        [filter = phaseFilter]
        [eis = [prefix.]attributeName[.suffix]
        [,[prefix.]attributeName[.suffix]]...]
        [allow-attr = allowAttributeList allow-target = allowTarget]
        [prune-attr = pruneAttributeList prune-target = pruneTargetList]
        [merge-dn-attr]
        [options = [ignore-from-result] [,result-required] [,prune-from-result]
        [,collapse-target]]
        )]
    }
```

viewName

Defines the name that the DSA command interpreter uses to identify the view. If the name contains spaces or non-alphanumeric characters, then it must be enclosed in quotes.

description = "*description*"

Describes the view. The description is any text string enclosed in quotes.

entry = *ViewDN*

Defines the base object of the view in LDAP format. This DN is the target of searches that invoke this view.

collapse-result

(Optional) Specifies that the view will merge all the results into one entry, which is the base-object of the search request invoking the view.

collapse-result-under-entry

(Optional) Specifies that the view will merge all the results into one entry, which is the entry DN returned by the phase one search. If the phase one search returns multiple entries then the view will be applied to each entry independently and multiple collapsed entries will be returned.

remap-originator

(Optional) Specifies that the originator, and hence access controls, are applied to the bind DN which is a virtual entry when binding to a view using a DN returned by a previous search with the 'collapse-result-under-entry'. The remap-originator option re-maps the originator to the underlying phase 1 entry allowing existing ACIs to be used.

view-entry-access-controls

(Optional) Specifies that temporary access to some sections of the view that are not visible to the user invoking the view are allowed. Use this in conjunction with 'trust-dsa-triggered-operations'. This works by ignoring access controls while the view searches are invoked and post-applying the access controls before the result is returned.

if (*condition*)

Specifies conditional views that must be met before the phase is performed. The conditional "if" and "else if" accept a view parameter a = (equals) or != (not equals) and a regular expression. The value substituted for the view parameter is compared to the regular expression.

Each phase/s can be conditional triggered based on information from previous phases. A condition consists of a views attribute an = or != and a regular expression. for example, "$2:userPassword=." will trigger the following brace enclosed phase/s if the phase 2 search result contains the userPassword attribute.

if (*condition*) { [*phase list*] }

else if (*condition*) { [*phase list*] }

else { [*phase list*] }

phase = *phaseNum*

Specifies the phase number.

A *phase* is a directory search within a view. A phase can use the results of previous phases in the same invocation of the view.

Each phase must be given a number, starting at one and incrementing by one for each subsequent phase.

Each phase includes the following parameters:

subtree = *phaseDN*

Defines the subtree of the search performed for this phase, in LDAP format. The subtree RDN elements can reference previous search phases. This can be omitted for attribute-level pruning/allowing.

For example:

- **"$1:dn"** - Searches with a subtree set to the DN of each search result returned in phase 1
- **"$1:dn[3]"** - Searches with a subtree set to the 3 top RDNs of the DN of each search result returned in phase 1
- **"uid=$1:uids,o=Democorp,c=AU"** - Replaces the last RDN of the subtree with the value/s of 'uids' returned by the phase 1 search. If multiple values were returned then multiple searches are performed.

eis = [*prefix*.]*attributeName*[.*suffix*] [,[*prefix*.]*attributeName*[.*suffix*]]...

(Optional) Defines the attributes that will be returned. Attribute names are separated by commas.

If *eis* is specified, then the phase returns only the specified attributes, plus any attributes referenced as parameters in later phases.

If *eis* is not specified, then all information items are returned.

If the search request that invoked the view specified an attribute name to be returned, then the attributes specified by *eis* are ignored.

Note: The *eis* does not need to include the linking attribute (the attribute required by a subsequent phase).

attributeName

Specifies an attribute to be returned. This follows the syntax for views parameters. <link>

as *attributeName*

Maps the result to this attribute. This is useful if the underlying data has naming conflicts.

Note: The syntax for both the attribute being returned and the mapped attribute must be identical.

prefix

Adds text to the start of the result. For more information, see Add a Prefix or Suffix to a Value in the *Administrator Guide*.

suffix

Adds text to the end of the result. For more information, see Add a Prefix or Suffix to a Value in the *Administrator Guide*.

scope = {subtree | base | one-level}

(Optional) Defines the scope of the search for the specified phase. The scope is one of the following:

subtree

Searches the entire subtree.

base

Searches the base object only.

one-level

Searches one level only.

filter = *phaseFilter*

(Optional for *base* and *one-level* searches) Defines the LDAP filter that the phase uses for its search. Any item in the filter can include views parameters.

If the scope of the search that invokes the view is base-object, then *filter* is ignored.

This LDAP search filter contains substitution items OR 'from-client'. When from-client the filter will directly reference the filter passed in be the client.

- **Phase 1 -** The substitution item must be $attrName. Where attrName is passed in on the subtree search on the view entry.
- **Phase >1 -** The substitution item must be a view attribute of the form $phase:attrName. $phase must reference the result from an earlier phase and attrName is the name of that attribute.

allow-attr = *allowAttributeList*

(Optional) Specifies the attributes that will be included in a phase result. This list of attributes follows the syntax for views parameters.

Specifies the attribute name whose value is appended to the list of attribute values in allow-target

allow-target = *allowTarget*

(Optional) Specifies the views parameter to take the value in allow-attr. If allow-target does not exist it is created when the first allow-attr is returned.

allowTarget follows the syntax for views parameters.

allow-target is used only with the allow-attr option.

prune-attr = *pruneAttributeList*

(Optional) Specifies the attributes that will be removed from a phase result.

pruneAttributeList follows the syntax for views parameters.

prune-attr is used only with the prune-target option.

prune-target = *pruneTargetList*

(Optional) Specifies the views parameter whose value is compared to the value of the attribute in prune-attr. If the two values match, then the attribute entry is removed from the result. Otherwise, the attribute entry is included in the result.

pruneTargetList follows the syntax for views parameters.

prune-target is used only with the prune-attr option.

merge-dn-attr

(Optional) Specifies the attribute of DN syntax that each DN of the current phase result will be returned under. For example, it is often useful to return the groups a user is a member of with the user's entry. If this is set to memberOf, the phase subtree is where the groups are stored and the filter = "member = $1:dn".

options = [ignore-from-result] [result-required] [prune-from-result] [collapse-target]

(Optional) Specifies a comma-separated list of processing that the phase should perform before it returns the results to the view. Possible options are as follows:

ignore-from-result

Specifies that the phase should return only those attributes whose values are referenced as parameters in later phases. If this is set, the phase search results are temporary. The results from this phase will not be returned to the client.

prune-from-result

Specifies that the phase should return only the DN and not return any attributes at all.

result-required

Specifies that the DSA should check if an attribute is referenced as a parameter in later phases does exist. If not, then the DSA aborts the search and raises an alarm.

collapse-target

(requires view option collapse-result-under-view-entry)

Instead of collapsing the view results under the entry specified by phase 1, this option allows for the view to be collapsed under a later phase. An error will occur if the later phase search returns multiple entries.

More information:

Views Parameters (see page 146)
clear view Command—Remove All View Definitions (see page 30)
get view Command—Display View Definitions (see page 44)

Views Parameters

You use a views parameter inside the *set view* command to specify a placeholder for a string. The DSA will determine the value of the views parameter when the view is invoked or after the DSA has run an earlier phase within the view.

The syntax of a views parameter is as follows:

`[$][phaseNumberList:]attributeName[substringIdentifier]`

$

(Optional) Specifies that the DSA should treat this as a views parameter.

If the $ is missing, then the string is treated as a literal attribute name and no substitution occurs. In that case, *substringIdentifier* must be blank.

***phaseNumberList*:**

(Optional) Defines the phase number that produces the value of *attributeName*. If *phaseNumber*: is omitted, then the DSA provides the value of *attributeName* from the filter given in the LDAP search command that invoked the view.

phaseNumber must be a smaller number than the number of the phase in which the parameter is used. That is, a phase can refer only to earlier phases of the view.

You can specify a list of phase numbers when using conditional views.

attributeName

Specifies the attribute whose value is used to replace the parameter.

substringIdentifier

(Optional) Specifies a section within an attribute value string.

There are two possible formats for the substring identifier.

The first form for *substringIdentifier* is useful if you want to extract a substring based on delimiter characters, and is as follows:

[*delimiterString*:*substringNumber*]

The square brackets and the colon are part of the syntax, and are required if this option is used.

delimiterString

Defines the delimiter inside the attribute value string. It is a literal string so, for example, it can include white space, quotation marks, and non-alpha-numeric characters. The delimiter string partitions the attribute value into substrings.

substringNumber

Specifies which substring to extract from the attribute value string. The substrings of the attribute value are numbered from left to right starting at one.

The second form for *substringIdentifier* is useful if you want to extract the leading DNs from a string of DNs, or if you want to extract the leading characters of the string. This form is as follows:

[*numberOfDNsOrCharacters*]

The square brackets are part of the syntax and are required if this option is used.

numberOfDNsOrCharacters

Specifies how many RDNs or characters to extract from the attribute value string, starting at the leftmost character.

If this parameter is used as part of the subtree specification of a phase ("subtree=..."), then it returns the number of RDNs; otherwise it specifies the number of characters.

Example: View Parameter $cn

When the view is invoked, the DSA replaces the example string with the value of the *cn* attribute provided in the filter of the search command that was used to invoke the view.

In this example, the search is invoked with the following command:

```
ldapsearch... (cn="Smith, John")
```

The *$cn* is replaced with the following value:

```
Smith, John
```

Example: View Parameter $cn[,:1]

When the view is invoked, the DSA replaces the example string with the characters preceding the first comma in the value of the cn attribute.

In this example, the search is invoked with the following command:

```
ldapsearch... (cn="Smith,John")
```

The *$cn[,:1]* is replaced with the following value:

```
Smith
```

Example: $cn[7]

When the view is invoked, the DSA replaces the example string with the first seven characters in the value of the cn attribute.

In this example, the search is invoked with the following command:

```
ldapsearch... (cn="Smith,John")
```

The *$cn[7]* is replaced with the following value:

```
Smith,J
```

Example: $3:cn

Immediately after running phase three of the view, the DSA replaces the example string with the value of the *cn* attribute returned by phase three. The example string is only valid in phases four and later.

Example: $2,3:cn

Using $2,3:cn in the following example uses $2:cn or $3:cn depending on which phase was invoked.

```
if ( condition )
{ (phase = 2) }
else
{ (phase = 3) }
```

Example: subtree= $1:dn

```
...

(phase=2
subtree = $1:dn
...
```

This is a fragment of a set view command. Immediately after running phase one, the DSA replaces the string $1:dn with the value of the DN of each search result returned in phase one. If phase one returns multiple DNs, then phase two is run multiple times—once for each DN result returned from phase one.

Example: subtree= $1:dn[3]

```
...

(phase=2
subtree = $1:dn[3]
...
```

This is a fragment of a set view command. Immediately after running phase one, the DSA replaces the string $1:dn[3] with the value of the first three RDNs of the DNs of each search result returned in phase one. If phase one returns multiple DNs, then phase two is run multiple times—once for each DN result returned from phase one.

Example: subtree="uid=$1:uids,o=Democorp,c=AU"

```
...

(phase=2
subtree="uid=$1:uids,o=Democorp,c=AU"
...
```

This is a fragment of a set view command. Immediately after running phase one, the DSA replaces the string $1:uid with the value of 'uids returned by the phase one search. If phase one returns multiple DNs, then phase two is run multiple times: once for each DN result returned from phase one.

For example, if phase returns 12345 for the value of uid, then this fragment is exactly equivalent to the following:

```
...
(phase=2
subtree="uid=12345,o=Democorp,c=AU"
...
```

shutdown Command

This command has the following format:

```
shutdown;
```

trace Command

The trace command displays details of service requests, confirmations, or errors. You need to specify the level of tracing you want to use.

trace full | none | trace-event-list ;

full

Traces everything. Full tracing degrades performance, so you should use it only during testing.

none

Turns off all tracing

trace-event-list

A comma-separated list of one or more of the following trace options:

alert

Displays authentication errors.

cert

Displays certificate operations.

connect

Displays connections.

diag

Traces local DXserver operations that were refused.

dsa

Similar to the x500 trace, but also includes tracing of the module flow inside the DSA.

error

Displays error messages of high severity. Compare with trace warn.

These events that may impact on the ability of DXserver to perform a requested operation. This is the default trace level.

We recommend that the error trace option is included in the set trace command during normal operation.

ldap

Traces detailed LDAP operations. The output can become quite large when searches return a large number of entries.

limit

Traces any violation of size or time limits.

query

Displays a one-line summary containing the server request and result.

stack

Displays detailed protocol tracing. The output can become quite large.

stats

Displays statistical information for each minute the DSA is not idle.

summary

Displays a one-line summary containing the service request and result.

time

Displays the time taken for successful operations.

To send this output to a separate file, use the set time-log command.

update

Displays update operations-add, delete, modify, and rename.

warn

Displays error messages of moderate severity. Compare with trace error.

Warn messages usually represent a user error, rather than a problem with DXserver.

x500

Displays the full details of the service request, confirmation, or error. This traces DAP, DSP, and LDAP operations. The output can become quite large when searches return a large number of entries.

trace disable assoc Command

The *trace disable assoc* command cancels the tracing of the requests/responses occurring on a specified binding.

This command has the following format:

```
trace disable assoc = binding-number;
```

trace enable assoc Command

The *trace enable assoc* command traces the requests/responses occurring on a specified binding. You will see no difference if used with x500 tracing.

Before you use the *trace enable assoc* command, use the *get users* command to find out the binding number.

This command has the following format:

```
trace enable assoc = binding-number;
```

binding-number

Specifies the binding that you want to trace

Example: trace enable assoc

This example shows how to disable all tracing, find out a binding number, and then trace all requests and responses occurring on that binding:

```
dsa> trace none;
trace none;
dsa> get users;
get users;

Association 2: connected for 34 seconds, idle for 25 seconds
Association 2: bound using LDAP (peer 155.35.131.63) as ANONYMOUS

Total Associations: 1
dsa> trace enable assoc = 2;
trace enable assoc = 2;
dsa>

<- #2 LDAP SEARCH-REQ
        invoke-id = 11   credit = 4
    Base object:
        <countryName "AU">
        <organizationName "Data">
        <organizationalUnitName "People">
        <commonName "Passadmin">
    Don't Search Aliases

-> #2 LDAP SEARCH-CONFIRM
        invoke-id = 11   credit = 1
    Entry:  (leaf)
        <countryName "AU">
        <organizationName "Data">
        <organizationalUnitName "People">
        <commonName "Passadmin">
    Contents:
        (objectClass inetOrgPerson, organizationalPerson, person, top)
        (userPassword "{SHA}hvfkN/qlp/zhXR3cuerq6jd2Z7g=")
        (commonName "Passadmin")
        (surname "Smithaa")
        (initials "aa")
```

unbind Command

The *unbind* command closes outgoing bindings to other DSAs. This command has the following format:

```
unbind all | dsa dsa-number ;
```

all

Unbinds all outgoing bindings from this DSA.

dsa *dsa-number*

Unbinds all outgoing bindings from this DSA to the specified DSA.

You can find the *dsa-number* by using the get dsas command.

More information:

abort Command—Close Bindings (see page 28)
get dsas Command—Display All Known Remote DSAs (see page 39)

update agreement Command

This command has the following format:

```
update agreement id.version;
```

id.version

The agreement identification and version numbers (for example, 1.2).

Chapter 3: Other Commands

This section contains the following topics:

DXadmind Commands

DXadmind is a background process that runs on each host that contains a DSA. DXmanager uses DXadmind to communicate with the DSAs. The DXadmind on each host collects information for DXmanager and manages the DSAs on that host on behalf of DXmanager.

The CA Directory installation process installs dxadmind, so dxadmind is started each time the operating system starts up.

If you are having problems with DXmanager, you may need to monitor, start, or stop DXadmind.

The syntax is as follows:

`dxadmind` *`command`*

command

Specifies the command and is one of the following:

debug [*level*] [local]

Start the service in debug mode.

help [local]

Display basic configuration information.

install *name*

Install the service instance specified.

remove [*name*]

Uninstall the dxadmind service on the local host.

setup *trustedhost port [password]*

Write an empty config file with the port, trusted host, and password.

start [*name*]

Start the dxadmind service.

status [*name*]

Display the status of the dxadmind service.

stop [*name*]

Stop the dxadmind service.

dxadmind debug Command—Display Debugging Information about DXadmind

The *dxadmind debug* starts DXadmind and displays debugging information on the screen. This can be useful to help you give more details to CA Technical Support.

This command has the following format:

```
dxadmind debug [level] [local]
```

level

Specifies the debug level

local

In local mode only one server is allowed in the XML configuration file and it is assumed to be the current host. Local mode is for situations where no network interfaces are available.

dxadmind help Command—Check the DXadmind Configuration

This command has the following format:

```
dxadmind help
```

The output includes the following information:

Config version

Lists the latest version of the configuration that DXadmind has received. You can compare this number with the version shown in DXmanager.

DXadmind Port

Lists the port on which DXadmind listens for requests from DXmanager. You can change this port using the *dxadmind setup* command.

DXadmind Trap Port

Lists the port on which DXadmind listens for SNMP traps from DSAs on the host.

DXmanager Host

Lists the IP address of the Management Server that DXadmind trusts. You can change this using the *dxadmind setup* command.

Configured DSAs

Lists the DSAs from the configuration that exist on this host.

Known local network addresses

Lists the interfaces that DXadmind can find on this host.

dxadmind start Command—Start DXadmind

The *dxadmind start* command starts DXadmind.

This command has the following format:

```
dxadmind start
```

dxadmind stop Command—Stop DXadmind

The *dxadmind stop* command stops DXadmind.

This command has the following format:

```
dxadmind stop
```

dxadmind setup Command—Change the DXadmind Configuration

The dxadmind setup command lets you configure DXadmind to recognize a DXmanager computer as a trusted management server. It writes a new configuration file with the port, trusted host, and password.

This command has the following format:

`dxadmind setup trustedhost port [password]`

trustedhost

Specifies the host computer that runs DXmanager.

Note: You can specify *trustedhost* using an IP address or using a host name. If you use the host name format, then you must have reverse DNS lookups enabled on the DNS.

port

Specifies the port that DXmanager will use to access DXadmind. This must be the same as the DXadmind port you specified in DXmanager.

password

(Optional) Specifies the password that DXmanager will use to access DXadmind. This must be the same as the DXadmind password you specified in DXmanager.

DXwebserver Commands

The Directory Management components use Tomcat, which is an open-source web server. The Tomcat web server is installed as CA Directory Web Server.

On Windows, use the Services window to control CA Directory Web Server. The DXwebserver commands are useful for UNIX only.

dxwebserver start Command

Valid on UNIX

The following command starts the web server:

`dxwebserver start.`

dxwebserver stop Command

Valid on UNIX

The following command stops the web server:

```
dxwebserver stop
```

dxwebserver status Command

Valid on UNIX

The following command displays the status of the web server:

```
dxwebserver status
```

Chapter 4: DXtools

The *DXtools* are a set of command-line utilities that come with CA Directory. These tools help you manage directory administration, work with LDIF data, load and unload data to and from a directory, and to extract and convert schemas for use with CA Directory.

This section describes the syntax and usage of each tool.

How to Use the DXtools

You can run the DXtools in the following ways:

- Run the DXtools commands on the host, using the DSA console.
- Run the DXtools commands on a remote host, using the DSA console over a TCP/IP network.
- Include the DXtools commands in your scripts.

All tools return zero on success and non-zero when an error occurs.

DXHOME Environment Variable

Some tools require that the DXHOME environment variable is set to the home path of DXserver. This is done automatically when CA Directory is installed.

Some tools expect the DSA configuration files to be located in the *config* folder under the path in DXHOME.

Lists of DXtools by Function

The following sections list the DXtools.

Tools to Manage the Datastore

- DXdumpdb Tool
- DXemptydb Tool (see page 177)
- DXloaddb Tool (see page 180)
- DXnewdb Tool (see page 189)

Tools for DSA Operations

These tools let you run operations on a running DSA.

These tools are slightly modified versions of the public domain LDAP tools (ldapdelete, ldapmodify, ldaprename, and ldapsearch):

- DXdelete Tool (see page 172)
- DXmodify Tool (see page 182)
- DXrename (see page 193)
- DXsearch Tool (see page 199)

Tools to Configure DSAa and Convert Schemas

This tool lets you generate and work with certificates:

- DXcertgen Tool

These tools simplify the extraction of schemas from LDAP directories, and the conversion of schemas into a format appropriate for CA Directory and for use by other DXtools:

- DXschemaldif Tool (see page 198)
- ldif2dxc Tool (see page 205)

Tools for Routine Administration

These tools help with routine directory administration:

- DXinfo Tool (see page 177)
- DXpassword Tool
- DXsyntax Tool (see page 204)
- Sample Tools

LDIF Tools

LDIF is a format suitable for describing directory information or changes to be made to directory information.

LDIF files are text files that store directory information in LDIF. You can use LDIF files to transfer directory information between LDAP directory servers or to describe a set of changes to be applied to a directory.

These tools are useful to create, manipulate, and use LDIF data:

- csv2ldif Tool (see page 164)
- ldifdelta Tool (see page 209)
- ldifsort Tool (see page 212)
- DXmodify Tool (see page 182)

Exit Status Codes for the DXtools

The DXtools share common exit codes, though not all exit codes apply to all tools. The exit codes are as follows:

0

Success

Non-zero value

An error condition. The cause is explained by an error message.

csv2ldif Tool—Create an LDIF File from a CSV File

Use the csv2ldif tool to create an LDIF file from a CSV file. You can then use the LDIF file as source input to DXmodify (see page 182).

csv2ldif directs its output to standard output. To create an LDIF file, redirect the output to a file name.

This command has the following format:

`csv2ldif options numfields LDTfile CSVfile`

options

Denotes one or more of the following options:

-b *badfile*

Specifies the output file name for CSV lines with bad format.

-d

Permits duplicate parent nodes to be generated.

-f *branching factor*

Specifies the number of branching factors. The default is 32.

Note: This is a low-level option. Do not use it unless you are told to by CA Technical Support.

-i *Numberoflines*

Ignores the first NumberofLines lines of the CSV file.

-s *sep*

Defines a field separator (default is a comma)

numfields

Specifies the total number of fields defined in the input CSV file.

LDTfile

Specifies the name of the LDT file.

CSVfile

Specifies the name of the input CSV file.

Example: Convert Sample CSV Data Using an LDT File

The following command uses the csv2ldif utility to convert CSV data to LDIF:

```
csv2ldif -i 1 7 acme.ldt acme.csv > acme.ldi
```

The output is redirected to the acme.ldi file. The following is a portion of acme.ldi:

```
dn: o=Acme, c=US
oc: organization
dn: ou=Administration, o=Acme, c=US
oc: organizationalUnit
dn: cn=Fred Jones, ou=Administration, o=Acme, c=US
oc: organizationalPerson
postalAddress: 11 Main Street $ Newtown
surname: Jones
title: Manager
telephonenumber: +1 (123) 456 7890
telephonenumber: +1 (987) 654 3210
dn: ou=Sales, o=Acme, c=US
oc: organizationalUnit
```

Telephonenumber appears twice because it is a multi-valued attribute

More information:

The Format of LDT files (see page 246)

DXcertgen Tool—Generate and Work with Certificates

CA Directory includes the DXcertgen tool for generating and working with certificates.

dxcertgen certs Command—Create DSA and User Certificates

The command dxcertgen certs creates DSA and user certificates, and signs these using a root certificate.

If DXcertgen can use a keystore, it stores the root certificate and the private key there. If it cannot use keystore, it stores the root certificate (but not the private key) in a file.

If DXcertgen uses a keystore to hold the root certificate then it re-uses that root certificate when it creates new DSA and user certificates.

If DXcertgen does not use a keystore, it creates a new root certificate each time it creates new DSA or user certificates, and invalidates all certificates it had previously created. It also deletes the private key of the root certificate it has just created, to ensure that no more certificates will be created from that key and so ensure the integrity of the encryption.

Note: DXcertgen uses a keystore if (and only if) it finds keystore software in JAVA_HOME/bin/keytool.

DXcertgen can only create DSA certificates for DSAs that already exist.

It always stores DSA certificates in DXHOME/ssld/personalities. It stores user certificates in a keystore if one is specified in the -c option, or in the path specified in the -p option.

This command has the following syntax:

dxcertgen [-a *rootalias*] [-c *cert-ks-path* -C *cert-ks-password*] [-d *days*] [-D *dsaname*] [-i *issuer*] -p *cert-file-path* [-P root*cert-pk-ks-password*][-s root*cert-ks-path* [-S root*cert-ks-password]*] [-u *users*] [-Z *algorithm*] certs

-a *rootalias*

Specifies the root key in the keystore. Only use this if you use a root certificate keystore.

Default: dxcertgen

-c *cert-ks-path* -C *cert-ks-password*

cert-ks-path

Specifies the path to a keystore to use to store the created user certificates given by the -u option.

Default: DXHOME/config/ssld/javakeystores

cert-ks-password

Specifies the existing password to access the user certificate keystore. You should have set this password when you created the keystore.

-d *days*

Specifies the number of days for which the certificate will be valid.

Default: 365

-D *dsaname*

Specifies the DSA for which a new certificate will be created. If you do not use this option, DXcertgen will create certificates for all DSAs.

Only use this if you use a root certificate keystore.

-i *issuer*

Specifies the name to use if you are creating a root certificate. DXcertgen generates a root certificate with the name given here. Usually this will be your company name.

Default: "CN=DXCertGenCA,O=DXCertGenPKI,C=AU"

-p *cert-file-path*

Specifies the path to a file to use to store created user certificates.

Default: DXHOME/config/ssld/personalities

-P *rootcert-pk-ks-password*

Specifies a password to protect the private key in the root certificate keystore. You set this password here.

-s *rootcert-ks-path* [-S *rootcert-ks-password*]

rootcert-ks-path

Specifies the path to the root certificate keystore.

Default: DXHOME/config/ssld/javakeystores

rootcert-ks-password

Specifies the password to access the root certificate keystore. You should have set this password when you created the keystore.

Default: changeit

-u *users*

Specifies an LDIF file that contains a list of users to create certificates for.

-Z *algorithm*

Specify which signature algorithm to use when generating the certificate, where *algorithm* is one of the following:

SHA1

(Default) Signs the certificate using the SHA-1 algorithm.

SHA224

SHA256

SHA384

SHA512

Example: Create DSA Certificates

The following command creates certificates for all currently configured DSAs:

```
dxcertgen certs
```

DXcertgen writes the certificates and associated private keys files for DSAs into the directory DXHOME/ssld/personalities.

Each DSA certificate and private key is created in a file with a name based on the DSA name, as follows:

```
dsa_name_in_lowercase.pem
```

The files are not password protected or encrypted in any way. They should be made secure as soon as possible.

Example: Generate Certificates for CA Directory Users

The following command creates user certificates and writes the certificates and private keys to the file system:

```
dxcertgen -u $USERS_PATH/users.ldif -p $TEMP_PATH/users certs
```

The private keys that are written out are not password protected or encrypted in any way. They should be made secure as soon as possible.

The user certificates will be generated from the same private key as the DSA certificates.

dxcertgen certreq Command—Create a Certificate Signing Request (CSR)

If you want to use a certificate from another Certificate Authority (CA), you can use DXcertgen to create a Certificate Signing Request (CSR), which you can send to the authority.

This command creates a CSR and a private key. You do not send the private key to the CA.

If the request is successful, the CA will send back a certificate that has been digitally signed with the CA's private key. (You can then use this certificate by using the command *dxcertgen certmerge*.)

The command *dxcertgen certreq* stores the CSR in DXHOME/config/ssld/dsaname.csr, and stores the private key in DXHOME/config/ssld/dsaname.key.

The command has the following syntax:

```
dxcertgen [-D dsaname] [-Z] certreq
```

-D *dsaname*

Specifies the DSA for which a CSR will be created. If you do not use this option, DXcertgen will create requests for all DSAs.

-Z *algorithm*

Specify which signature algorithm to use when generating the certificate signing request, where *algorithm* is one of the following:

SHA1

(Default) Creates the private key using the SHA-1 algorithm.

SHA224

SHA256

SHA384

SHA512

dxcertgen certmerge Command—Use a CSR Response to Create a Certificate

When you have received a DSA certificate from a CA in response to a CSR, use this command to create the combination of DSA certificate and private key that a DSA needs.

The command *dxcertgen certmerge* does the following:

1. It verifies the received certificate.
2. It checks that the received certificate is, in fact, a response to the CSR that you sent using the dxcertgen certreq command.
3. It merges the returned certificate and the original private key and stores the DSA certificate and associated private key in the file *DXHOME/config/ssld/personalities/dsaname*.pem.

The command has the following syntax:

```
dxcertgen -D dsaname -n cert-file certmerge
```

-D ***dsaname***

Specifies the DSA for which a new merged certificate will be created.

-n *cert-file*

Specifies the certificate file that was returned from the external CA in response to the CSR.

dxcertgen importca Command—Import a Certificate

If you have a root certificate created by another Certificate Authority (CA), then you need to import the certificate into the file trusted.pem, so that DXserver will trust the certificate. You can use the *dxcertgen importca* command to do this.

DXcertgen can import certificates from the following formats:

- PEM (base-64 encoded)
- DER/CER ASN.1 (base-64 encoded or binary)
- PKCS#12

This command has the following syntax:

```
dxcertgen -n cert-file importca
```

-n *cert-file*

Specifies the certificate file that you want to add to the list of trusted certificates.

dxcertgen listca Command—List Root Certificates

The *dxcertgen listca* command displays a list of the certificates stored in *trusted.pem*.

Each certificate is uniquely numbered.

This command has the following syntax:

```
dxcertgen listca
```

dxcertgen removeca Command—Delete Root Certificates

This command removes the certificate matching the supplied certificate number and backs up the existing *trusted.pem* file into *trusted.pem.bak*.

Note: Certificate numbers will change when adding and removing certificates in trusted.pem. Be sure to perform a 'listca' prior to any certificate removal.

This command has the following syntax:

```
dxcertgen -r certnumber removeca
```

-r *certnumber*

Specifies the number corresponding to the certificate you want to delete.

dxcertgen report Command—Report on Certificates

You can use the DXcertgen tool to generate a report on the certificates.

The report summarizes all of the certificates, including root certificates, DSA personality certificates, and user certificates.

Each summary lists the following information:

- Subject of the certificate
- The Certificate Authority who issued the certificate
- Validity dates
- Whether the certificate is currently valid

This command has the following syntax:

```
dxcertgen report
```

DXdelete Tool—Delete Directory Entries

Use the DXdelete tool to delete one or more directory entries. To delete a single entry, supply the target DN identification by a direct command-line entry. To delete multiple entries, use input from a file.

This command has the following format:

`dxdelete [options] [dn]`

options

Denotes one or more of the following options:

-c

Runs in continuous mode. Errors are reported, but the process is not stopped.

-d *level* [-d *level*...]

Sets the LDAP debug levels.

level

Defines the level of debugging as follows:

-1	Enable all debugging
0	No debugging
1	Trace function
2	Debug packet handling
4	Heavy trace debugging
8	Connection management
16	Print out packages sent and received
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Stats log connections/operations/results
512	Stats log entries sent
1024	Print communication with shell backends
2048	Print entry parsing debugging

You can add numbers together to specify multiple debug levels at the same time. For example, a debug level of 6 specifies the debugging levels of both 2 and 4.

-D *bindDN*

Specifies the distinguished name of the user performing the bind.

-f *filename*

Specifies a file to read from, rather than standard input.

-h *dap-host*

Specifies the address or host name of the directory host. If you do not specify this, the tool uses *localhost* instead.

You can include OSI addressing for transport, session, and presentation SAPs by fully expanding *dap-host*:

```
hostname:port/tsel/ssel/psel
```

You can include binary and ASCII characters in the tsel, ssel, and psel selectors, using the % followed by the two hexadecimal digits that represent the ASCII code for the character, for example:

- / is expressed as %2F
- % is expressed as %25

-l *timelimit*

Specifies the time limit in seconds for each DAP operation.

-n

Shows what would be done, but does not actually do it. Use with the -v option for debugging.

-p *dap-port*

Specifies the port on directory host computer. If you do not specify this, the tool uses port 102, the OSI port, buy default

You can combine the -h and -p arguments into a single argument, and express them as a dotted IP address or hostname. For example, you can replace the options on the first line with those on the second:

```
-h 192.168.19.202 -p 19389
```

```
-h 192.168.19.202:19389
```

-v

Runs in verbose mode.

-w *password*

Specifies the bind password, which is used for simple authentication.

-Z [*ssld_config_filename*]

Specifies that the tool should start a TLS request.

Use -ZZ to require a successful response from the DSA.

ssld_config_filename

Specifies the name of the configuration file used by the -Z option. The default file is:

```
DXHOME/config/ssld/dxldap.conf
```

This configuration file is a text file. It must contain a line starting TLS_CACERT, and can optionally contain a second line starting TLS_REQCERT. Its format is as follows:

```
TLS_CACERT trusted_pem_file
[TLS_REQCERT {allow|demand|hard|never|try}]
```

In the configuration file, the lines have the following meaning:

TLS_CACERT *trusted_pem_file*

Specifies the file that contains certificates for all of the Certificate Authorities the client will recognize.

trusted_pem_file

Species the trusted pem file.

This must be an absolute reference to a full path, without environment variables.

Do not enclose trusted_pem_file in quotes.

[TLS_REQCERT {allow|demand|hard|never|try}]

Specifies what checks to perform on server certificates in a TLS session, if any.

If this line is missing, the system uses TLS_REQCERT demand

The keywords have the following meaning:

- allow - The client will request a server certificate and if no certificate is provided, the session proceeds normally. If a bad certificate is provided, it will be ignored and the session proceeds normally.
- demand - The client will request a server certificate and if no certificate is provided, or a bad certificate is provided, the session is immediately terminated. This is the default setting.
- hard - This is a synonym for *demand.*
- never - The client will not request or check any server certificate.
- try - The client will request a server certificate and if no certificate is provided, the session proceeds normally. However, if a bad certificate is provided, the session immediately terminates.

Example: dxldap.conf file on a Windows System, Specifying the TLS_REQCERT Setting

```
TLS_CACERT  c:\program files\CA\Directory\dxserver\config\ssld\trusted.pem
TLS_REQCERT allow
```

Example: dxldap.conf file on a UNIX System, Using the Default TLS_REQECRT Setting

```
TLS_CACERT  /opt/CA/Directory/dxserver/config/ssld/trusted.pem
```

dn

Specifies the distinguished name of entry to delete.

Example: Delete an Entry

This example uses the Democorp sample directory supplied with CA Directory. You can repeat this example as a training exercise.

The following command deletes the entry Murray J Horsfall:

```
dxdelete -v -h hostname:19389 "cn=Murray J HORSFALL,ou=Repair,
ou=Operations,o=Democorp,c=AU"
```

To test that the entry was deleted, use the DXsearch tool.

DXdisp Tool—Clear or List the Update Time for Multiwrite-DISP Replications

Use the DXdisp tool to clear the last update time, when using multiwrite-DISP.

When multi-write-disp-recovery is set to TRUE, and you reload a DSA or remove a DSA from the multiwrite-DISP configuration, you must clear the last update time.

This command reads the configuration file of every DSA on the current machine and removes all references in the DISP agreements to the DSA.

Note: For a list of the status codes returned by all the DXtools commands, including this command, see Exit Status Codes for the DXtools (see page 163).

This command has the following format:

```
dxdisp DSA name
```

DSA

Specifies the DSA.

DXdumpdb Tool-

Use the DXdumpdb tool to export data from a datastore to an LDIF file.

Note: For a list of the status codes returned by all the DXtools commands, including this command, see Exit Status Codes for the DXtools (see page 163).

This command has the following format:

```
dxdumpdb options DSA
```

options

Denotes one or more of the following options:

-f *filename*

Specifies the file to receive the exported data. If this option is not specified, the output goes to standard output or the screen.

-i

Specifies to include listed attributes.

-v

Runs in verbose mode.This option switches on error and status tracing. For the -v option to work, you must also specify the -f option.

-x

Specifies to exclude listed attributes.

Note: You should not exclude attributes that form part of the DN.

-z

Specifies that DXdumpdb dumps from the copy of the datastore that is produced by the console command *dump dxgrid-db*.

DSA

Defines the DSA. DXdumpdb looks in the configuration files of this DSA to find the datastore to export to an LDIF file.

Example: Extract Democorp Data to Screen

The following example prints the LDIF format data from the datastore of the *democorp* DSA to the screen:

```
dxdumpdb democorp
```

DXemptydb Tool—Delete All Data from a Datastore

Use the DXxemptydb tool to remove all the data from a datastore. DXemptydb does not destroy the datastore, but leaves it ready to accept new data.

Note: For a list of the status codes returned by all the DXtools commands, including this command, see Exit Status Codes for the DXtools (see page 163).

This command has the following format:

```
dxemptydb DSA
```

DSA

Specifies the name of the DSA whose datastore is to be emptied.

DXextenddb Tool—Increase the Size of a Datastore

Use the DXextend tool to increase the size of a specified data store.

The size of the datastore is set in the configuration file. If you want to change the size, change the configuration file and then run the DXextenddb tool.

This tool has the following syntax:

```
dxextenddb dsaname
```

dsaname

Defines the name of the DSA you want to extend.

DXinfo Tool—Collect System Information

DXinfo collects information about your system and saves this information in a file. You can then send this file to CA to help our support staff with any problems you are having.

Output from DXinfo might contain sensitive information. To avoid having this information sent to CA Technologies, you can do the following:

- Specify which information you do not want collated.
- Edit the files after they have been collated, to remove sensitive information.

This command has the following format:

`dxinfo [-f filename] [-x info-list] [-v] [-?]`

-f *filename*

Specify the file name for the system information. If you do not specify a file name, DXinfo stores the information in etrdir_dxinfo.log.

-x *info-list*

Comma-separated list of information to be excluded, as follows:

- logs – Exclude the log files
- config – Exclude the configuration files
- sysdata – Exclude system information

-v

Runs in verbose mode.

-?

Displays usage information for this tool

Default DXinfo File Names and Locations

DXinfo saves the following information in these log files. They are in the location in which you ran the DXinfo tool:

System information

cadir_dxinfo.log

DSA logs

Windows: cadir_logs.cab

UNIX: cadir_logs.tar.Z

DSA configuration files

Windows: cadir_config.cab

UNIX: cadir_config.tar.Z

Information Collated by the DXinfo Tool

This topic describes the information that the DXinfo tool collates.

Version of the DXinfo tool

The version of the DXinfo tool and the time you ran the tool

Version of CA Directory

The output from the command *dxserver version*

For Windows only, the %DXHOME%\bin\dxserver.exe file version information.

Version of the operating system

On Windows: The information found with the *winver* command

On UNIX: The information found with the *uname -a* command

Installation location for CA Directory

The value of DXHOME

User environment

On Windows: The information found with the *set* command, for both the current and system users

Configuration files

Information about the files in the *config* directory.

Log files

Information about the files in the *logs* directory.

DXloaddb Tool—Load a Datastore from an LDIF File

Use DXloaddb to load a datastore from a LDIF file. The datastore must already exist. All previous information in the datastore is deleted.

Usage notes:

- The LDIF file does not need to be sorted.
- DXloaddb hashes any password entry in the LDIF file that is in clear text.

  If a hash algorithm is specified in the DSA configuration, DXloadbdb uses that. Otherwise it uses SHA-1.
- By default, DXloaddb uses the DSA's configuration for operational attribute handling:
  - If *op-attrs = true* then any operational attributes in the LDIF file are loaded into the datastore.

    Any entries in the LDIF file that do not have a createTimestamp, have a creatTimestamp added to the datastore.
  - If *op-attrs = false* then operational attributes in the LDIF file are ignored and no operational attributes are created by the DXloaddb.

This command has the following format:

`dxloaddb [options] dsa ldif-file`

options

Denotes one or more of the following options:

-n

Specifies that DXloaddb does not do any actions.

-O

Specifies that DXloaddb includes standard operational attributes, such as password policy (for example, number of login attempts), and time stamp attributes. If this option is specified, DXloaddb creates any operational attributes that are not defined in the LDIF file.

-s

Specifies that DXloaddb produces the following statistics concerning the datastore:

- Total data size in MB
- Total number of entries
- Number of entries ignored
- Amount of padding in the datastore file in KB
- Average number of entries per MB

-v

Specifies verbose output.

ldif-file

The name of the LDIF file to load into the datastore.

DSA

Defines the DSA whose datastore is to be loaded.

Example: Create and Load a Datastore

The correct sequence in which to create and load a datastore is:

```
dxnewdb
dxloaddb
```

Example: Load LDIF Data into Datastore

The following example loads the data from democorp.ldif file to datastore democorp:

```
dxloaddb democorp democorp.ldif
```

IThe following is a possible part of democorp.ldif :

```
dn: o=Democorp, c=US
oc: organization
dn: ou=Administration, o=Democorp, c=US
oc: organizationalUnit
dn: cn=Fred Jones, ou=Administration, o=Democorp, c=US
oc: organizationalPerson
postalAddress: 11 Main Street $ Newtown
surname: Jones
title: Manager
telephonenumber: +1 (123) 456 7890
telephonenumber: +1 (987) 654 3210
dn: ou=Sales, o=democorp, c=US
oc: organizationalUnit
```

Telephonenumber appears twice because it is a multi-valued attribute

DXmodify Tool—Add New or Changed Information to a Directory

Use the DXmodify tool to do any of the following:

- Populate an empty directory
- Add complete new entries to an existing directory
- Add new attributes to existing entries
- Modify, rename, or delete attributes of an entry
- Load a binary file

When you run the DXmodify tool, you can enter the new or changed information from standard input or from an LDIF file.

This command has the following format:

```
dxmodify [action] [options]
```

action

Specifies the action the tool will take. The DXmodify tool follows any changetype directives in the LDIF file. If the LDIF file contains entries and no changetype directives, then you can tell DXmodify whether it should add these entries or replace existing entries. DXmodify replaces entries if either of the following conditions are true:

- The entry in the LDIF file is a modify operation and the modify type is specified for an attribute in the LDIF file
- -r is specified

-a

Adds the entries in the LDIF file to the directory.

-r

Replaces the entries or attributes in the directory with those in the LDIF file.

options

Denotes one or more of the following options:

-c

Runs in continuous mode. Errors are reported, but the process is not stopped.

-d *level* [-d *level*...]

Sets the LDAP debug levels.

level

Defines the level of debugging as follows:

-1	Enable all debugging
0	No debugging
1	Trace function
2	Debug packet handling
4	Heavy trace debugging
8	Connection management
16	Print out packages sent and received
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Stats log connections/operations/results
512	Stats log entries sent
1024	Print communication with shell backends
2048	Print entry parsing debugging

You can add numbers together to specify multiple debug levels at the same time. For example, a debug level of 6 specifies the debugging levels of both 2 and 4.

-D *bindDN*

Specifies the distinguished name of the user performing the bind.

-f *filename*

Specifies a source LDIF file. If you do not specify this option, or do not specify a file name, then DXmodify waits for input from standard input.

-h *dap-host*

Specifies the address or host name of the directory host. If you do not specify this, the tool uses *localhost* instead.

You can include OSI addressing for transport, session, and presentation SAPs by fully expanding *dap-host*:

hostname:port/tsel/ssel/psel

You can include binary and ASCII characters in the tsel, ssel, and psel selectors, using the % followed by the two hexadecimal digits that represent the ASCII code for the character, for example:

- / is expressed as %2F
- % is expressed as %25

-l *timelimit*

Specifies the time limit (in seconds) for each DAP operation.

-n

Shows what would be done, but does not actually do it. This can be useful for debugging purposes, usually you use this with the -v option.

-p *dap-port*

Specifies the port on directory host computer. If you do not specify this, the tool uses port 102, the OSI port, buy default

You can combine the -h and -p arguments into a single argument, and express them as a dotted IP address or hostname. For example, you can replace the options on the first line with those on the second:

```
-h 192.168.19.202 -p 19389
```

-h 192.168.19.202:19389

-q

Runs in quiet mode, in which successful operations are not reported.

-s *time*

Specifies the time (in milliseconds) to sleep after each operation.

-v

Runs in verbose mode.

-w *password*

Specifies the bind password, which is used for simple authentication.

-Z [*ssld_config_filename*]

Specifies that the tool should start a TLS request.

Use -ZZ to require a successful response from the DSA.

ssld_config_filename

Specifies the name of the configuration file used by the -Z option. The default file is:

```
DXHOME/config/ssld/dxldap.conf
```

This configuration file is a text file. It must contain a line starting TLS_CACERT, and can optionally contain a second line starting TLS_REQCERT. Its format is as follows:

```
TLS_CACERT trusted_pem_file
[TLS_REQCERT {allow|demand|hard|never|try}]
```

In the configuration file, the lines have the following meaning:

TLS_CACERT *trusted_pem_file*

Specifies the file that contains certificates for all of the Certificate Authorities the client will recognize.

trusted_pem_file

Species the trusted pem file.

This must be an absolute reference to a full path, without environment variables.

Do not enclose trusted_pem_file in quotes.

[TLS_REQCERT {allow|demand|hard|never|try}]

Specifies what checks to perform on server certificates in a TLS session, if any.

If this line is missing, the system uses TLS_REQCERT demand

The keywords have the following meaning:

- allow - The client will request a server certificate and if no certificate is provided, the session proceeds normally. If a bad certificate is provided, it will be ignored and the session proceeds normally.
- demand - The client will request a server certificate and if no certificate is provided, or a bad certificate is provided, the session is immediately terminated. This is the default setting.
- hard - This is a synonym for *demand.*
- never - The client will not request or check any server certificate.
- try - The client will request a server certificate and if no certificate is provided, the session proceeds normally. However, if a bad certificate is provided, the session immediately terminates.

Example: dxldap.conf file on a Windows System, Specifying the TLS_REQCERT Setting

```
TLS_CACERT  c:\program files\CA\Directory\dxserver\config\ssld\trusted.pem
TLS_REQCERT allow
```

Example: dxldap.conf file on a UNIX System, Using the Default TLS_REQECRT Setting

```
TLS_CACERT  /opt/CA/Directory/dxserver/config/ssld/trusted.pem
```

Example: Make Multiple Changes to an Entry

This example uses the Democorp sample directory supplied with CA Directory. You can repeat this example as a training exercise.

You can make multiple changes, such as changing the title and postal address, adding a second telephone number, and deleting the description of an entry.

This example shows that you can replace the values of multiple attributes using one replace statement as long as the replace statement specifies the first attribute name in the series.

1. Create an LDIF file named *h-modify.ldif* that contains the following:

```
dn: cn=Murray HORSFALL, ou=Repair,ou=Operations,o=Democorp,c=AU
changetype: modify
replace: title
title: Chief Information Officer
-
add: telephone
telephone: 797 8888
-
delete: description
-
replace: postalAddress
postalAddress: 173 Toorak Rd $ South Yarra
postalCode: 3066
```

2. Use DXmodify to apply the edited file as follows:

```
dxmodify -h localhost:19389 -f h-modify.ldif
```

Example: Add a Binary File

This shows how to add a JPEG file with a personnel record from staff.ldif.

For JPEG files, the object class is *cosinePilotObject*, the X.500 attribute name is *cosineJpegPhoto*, and the LDAP attribute name is *JpegPhoto*.

This example uses the Democorp sample directory supplied with CA Directory. You can repeat this example as a training exercise.

To add a binary file, follow these instructions:

1. Decide on the directory schema object class and attribute to use to hold the binary data.

   For this example, use the *cosineJpegPhoto* attribute within the *cosinePilotObject* object class.

2. Create entries in an LDIF file with the following syntax:

   ```
   attributeName:< FILE://path
   ```

   For this example, create staff.ldif with the following form:

   ```
   dn: cn=Peter Bell,ou=Infrastructure,ou=Support,o=Democorp,c=AU
   oc: organizationalPerson
   oc: newPilotPerson
   oc: cosinePilotObject
   cn: Peter Bell
   sn: BELL
   cosineJpegPhoto:< FILE://d:\temp\PHOTO\BELPE01.jpg
   title: Design Supervisor
   telephone: 881 9256
   description: Computing
   mail: Peter.BELL@Democorp.com
   postalAddress: 7-11 Fine Street$Penville CA
   postalCode: 32750
   ```

3. Run the following command:

   ```
   dxmodify -a -c -h hostname:19389 -f staff.ldif
   ```

DXnewdb Tool—Create a New Datastore

The DXnewdb tool creates a new datastore.

If the DSA is running or if a datastore already exists for the DSA, DXnewdb will exit immediately.

The size of the datastore is set in the configuration file.

Note: For a list of the status codes returned by all the DXtools commands, including this command, see Exit Status Codes for the DXtools (see page 163).

This command has the following format:

dxnewdb *DSA-name*

DSA-name

Defines the DSA.

DXnewdsa Tool—Create a New DSA

Use the DXnewdsa tool to create a new DSA.

If a DSA with the same name already exists, the command exits.

If the datastore exists, DXnewdsa uses it, without changing its size or contents. If the datastore does not exist, DXnewdsa creates the datastore.

DXnewdsa creates configuration files for the new DSA. The configuration includes the following commands:

```
#gxgrid configuration
set dxgrid-db-location = ...;
set dxgrid-db-size = ...;
set cache-index = all-attributes;
set lookup-cache = true;
```

Note: For a list of the status codes returned by all the DXtools commands, including this command, see Exit Status Codes for the DXtools (see page 163).

DXnewdsa has the following format:

```
dxnewdsa  [options] dsaname port [prefix]
```

options

Denotes one or more of the following options:

-t {data | router}

data

(default) Specifies that DXnewdsa should create a data DSA.

router

Specifies that DXnewdsa should create a router DSA.

-l *location*

(Optional) Defines the directory that holds the datastore. This is only valid for a data DSA.

The default directory is DXHOME/data.

If the location is provided as a relative location, DXnewdsa takes this as relative to the current directory, to produce an absolute path. DXnewdsa uses the -l parameter to create a command in the DSA configuration file, as follows:

```
set dxgrid-db-location = location;
```

-s *size*

(Optional) Defines the size of the datastore in MB. This is only valid for a data DSA. It is ignored if the datastore already exists.

DXnewdsa writes the size as a dxgrid-db-size setting in the DSA configuration file.

The maximum is 60,000.

The default is 500 MB.

dsaname

Defines the name of the DSA. Maximum length is 31 characters.

port

Specifies the port number of the DSA.

prefix

(Optional) Defines the namespace prefix. To specify the prefix, use LDAP syntax; for example as follows:

```
ou=Internet Sales, o=ACME Corp, c=US
```

Example: Create a data DSA with Default Options

```
dxnewdsa DSA1 1234
```

This command creates a data DSA1 and specifies port 1234 and a null prefix.

Example: Create a Data DSA with Simple Prefix

```
dxnewdsa DSA1 1234 ou=Internet Sales, o=ACME Corp, c=US
```

The configuration includes the prefix setting in the knowledge file in the following form:

```
set dsa DSA1 =
{
    ...
    prefix = <c US> <o Acme Corp> <ou Internet Sales>
    ...
}
```

DXpassword Tool—Hash a Password

The DXpassword tool lets you hash a password. You can then add the hashed password to a knowledge file.

This command has the following format:

```
dxpassword [-v] [-P algorithm] password
```

-P *algorithm*

Hashes the password, where *algorithm* is one of the following:

SHA

(Default) Hashes the password using the SHA-1 algorithm

SSHA

Hashes the password using the Salted SHA-1 algorithm. This algorithm produces a different hash even for the same clear text password, which is more secure.

SHA512

Hashes the password using the SHA-512 algorithm

CRYPT

Hashes the password using the UNIX crypt method.

MD5

Hashes the password using the Message Digest algorithm

CADIR

Hashes the password using a reversible obfuscation algorithm. Use this option to hash a password before including it in a knowledge file in the *dsa-password* or *ldap-dsa-password* configuration items. This protects the password from users with access to the computer running the DSA.

-v

Runs in verbose mode.

password

Specifies the password to be hashed.

Example: Hashing a Password Using MD5

The following command hashes the password *qwer123* using the MD5 algorithm:

```
dxpassword -P MD5 qwer123
```

The output from this command is:

```
{MD5}VT6DymlpOzPvc5WMBLejFQ==
```

Example: Verbose Output from the DXpassword Tool

The following command hashes the password *qwer123* using the SHA-1 algorithm, which is the default:

```
dxpassword -v qwer123
```

The output from this command is:

```
'qwer123' encoded is: {SHA}ACldbAY9DZzMbi3DJJGYppRIayY=
```

DXrename Tool—Change the Name of a Directory Entry

Use the DXrename tool to change the distinguished name of a directory entry. The utility can source the rename data for a single entry from the keyboard or for multiple entries from a file.

This command has the following format:

```
dxrename [options] [dn newdn]
```

options

Denotes one or more of the following options:

-c

Runs in continuous mode. Errors are reported, but the process is not stopped.

-d *level* [-d *level*...]

Sets the LDAP debug levels.

level

Defines the level of debugging as follows:

-1	Enable all debugging
0	No debugging
1	Trace function
2	Debug packet handling
4	Heavy trace debugging
8	Connection management
16	Print out packages sent and received
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Stats log connections/operations/results
512	Stats log entries sent
1024	Print communication with shell backends
2048	Print entry parsing debugging

You can add numbers together to specify multiple debug levels at the same time. For example, a debug level of 6 specifies the debugging levels of both 2 and 4.

-D *bindDN*

Specifies the distinguished name of the user performing the bind.

-f *filename*

Specifies a file to read from, rather than standard input.

-h *dap-host*

Specifies the address or host name of the directory host. If you do not specify this, the tool uses *localhost* instead.

You can include OSI addressing for transport, session, and presentation SAPs by fully expanding *dap-host*:

hostname:port/tsel/ssel/psel

You can include binary and ASCII characters in the tsel, ssel, and psel selectors, using the % followed by the two hexadecimal digits that represent the ASCII code for the character, for example:

- / is expressed as %2F
- % is expressed as %25

-l *timelimit*

Specifies the time limit in seconds for each DAP operation.

-n

Shows what would be done, but does not actually do it. Use with the -v option for debugging.

-p *dap-port*

Specifies the port on directory host computer. If you do not specify this, the tool uses port 102, the OSI port, buy default

You can combine the -h and -p arguments into a single argument, and express them as a dotted IP address or hostname. For example, you can replace the options on the first line with those on the second:

```
-h 192.168.19.202 -p 19389
```

```
-h 192.168.19.202:19389
```

-r

Removes the old RDN.

-v

Runs in verbose mode.

-w *password*

Specifies the bind password, which is used for simple authentication.

-Z [*ssld_config_filename*]

Specifies that the tool should start a TLS request.

Use -ZZ to require a successful response from the DSA.

ssld_config_filename

Specifies the name of the configuration file used by the -Z option. The default file is:

```
DXHOME/config/ssld/dxldap.conf
```

This configuration file is a text file. It must contain a line starting TLS_CACERT, and can optionally contain a second line starting TLS_REQCERT. Its format is as follows:

```
TLS_CACERT trusted_pem_file
[TLS_REQCERT {allow|demand|hard|never|try}]
```

In the configuration file, the lines have the following meaning:

TLS_CACERT *trusted_pem_file*

Specifies the file that contains certificates for all of the Certificate Authorities the client will recognize.

trusted_pem_file

Species the trusted pem file.

This must be an absolute reference to a full path, without environment variables.

Do not enclose trusted_pem_file in quotes.

[TLS_REQCERT {allow|demand|hard|never|try}]

Specifies what checks to perform on server certificates in a TLS session, if any.

If this line is missing, the system uses TLS_REQCERT demand

The keywords have the following meaning:

- allow - The client will request a server certificate and if no certificate is provided, the session proceeds normally. If a bad certificate is provided, it will be ignored and the session proceeds normally.
- demand - The client will request a server certificate and if no certificate is provided, or a bad certificate is provided, the session is immediately terminated. This is the default setting.
- hard - This is a synonym for *demand.*
- never - The client will not request or check any server certificate.
- try - The client will request a server certificate and if no certificate is provided, the session proceeds normally. However, if a bad certificate is provided, the session immediately terminates.

Example: dxldap.conf file on a Windows System, Specifying the TLS_REQCERT Setting

```
TLS_CACERT  c:\program files\CA\Directory\dxserver\config\ssld\trusted.pem
TLS_REQCERT allow
```

Example: dxldap.conf file on a UNIX System, Using the Default TLS_REQECRT Setting

```
TLS_CACERT  /opt/CA/Directory/dxserver/config/ssld/trusted.pem
```

dn

Specifies the distinguished name of the entry that is to be renamed.

newrdn

Specifies the new RDN.

Example: Change Middle Initial

This example uses the Democorp sample directory supplied with CA Directory. You can repeat this example as a training exercise.

Consider an example entry Murray Horsfall

1. Change the RDN to insert a middle initial J.

   The example uses the input file option. The example file name is *filename.txt*:

   ```
   cn=Murray HORSFALL,ou=Repair,ou=Operations,o=Democorp,c=AU
   cn=Murray J HORSFALL
   ```

2. Use the DXrename tool as follows:

   ```
   dxrename -r -h hostname:1900 -f filename.txt
   ```

3. Use the DXsearch tool to check the results of the rename:

   ```
   dxsearch -L -h hostname:19389 "(sn=horsfall)"

   dn: cn=Murray J HORSFALL,ou=Repair,ou=Operations,o=Democorp,c=AU
   oc: organizationalPerson
   oc: newPilotPerson
   oc: 0.9.2342.19200300.99.3.2
   cn: Murray J HORSFALL
   sn: HORSFALL
   title: Chief Information Officer
   telephone: 797 8877
   telephone: 797 8888
   mail: Murray.HORSFALL@Democorp.com
   postalAddress: 173 Toorak Rd $ South Yarra
   postalCode: 3066
   ```

DXschemaldif Tool—Extract the Schema from an LDAP Directory

Use the DXschemaldif tool to extract the schema from an external LDAP directory. The schema is extracted into LDIF format, and can be directed to the screen or a file.

This command has the following format:

```
dxschemaldif [-v] [options] hostaddr:port [> filename]
```

options

Denotes one or more of the following options:

-D *binddn*

Specifies the bind DN. If you don't specify the bind DN, the tool will bind anonymously.

-v

Runs in verbose mode.

-w *password*

Specifies the password (for simple authentication only).

hostaddr:*port*

Specifies the host address and port of the LDAP server (such as DXserver).

> *filename*

Redirect the output from the screen to the named file.

Note: To get help on the tool, enter the command *dxschemaldif* with neither options nor parameter.

Example: Extract Schema from V3 LDAP

To extract the schema from a Version 3 LDAP directory and redirect the output to the new_schema.ldif file, enter the following command:

```
dxschemaldif -v ldapserver:389 > new_schema.ldif
```

You can use the ldif2dxc tool to convert the LDIF schema into the CA Directory schema format.

DXsearch Tool—Search a Directory

Use the DXsearch tool to search within a specified directory using defined filters. The utility lets you specify search output as LDIF or text, or can write each returned attribute to a file.

This command has the following format:

dxsearch [*options*] *filter* [*attributelist*]

options

Denotes one or more of the following options:

-a { never | always | search | find }

Sets alias dereferencing.

-A

Retrieves attribute names only (no values).

-b *basedn*

Specifies the base DN for the search.

-B

Does not suppress printing of non-ASCII values.

-c

Runs in continuous mode. Errors are reported, but the process is not stopped.

-d *level* [-d *level*...]

Sets the LDAP debug levels.

level

Defines the level of debugging as follows:

-1	Enable all debugging
0	No debugging
1	Trace function
2	Debug packet handling
4	Heavy trace debugging
8	Connection management
16	Print out packages sent and received
32	Search filter processing
64	Configuration file processing
128	Access control list processing
256	Stats log connections/operations/results
512	Stats log entries sent
1024	Print communication with shell backends
2048	Print entry parsing debugging

You can add numbers together to specify multiple debug levels at the same time. For example, a debug level of 6 specifies the debugging levels of both 2 and 4.

-D *bindDN*

Specifies the distinguished name of the user performing the bind.

-f *filename*

Specifies a file to read from, rather than standard input.

-h *host*

Specifies the directory host. If you do not specify this, the tool uses *localhost* instead.

-H *LDAP_URI*

Specifies the LDAP URI of the directory host. If you do not specify this, the tool uses *localhost* instead.

You can use an IPv6 address, as in the following example:

```
-H ldap://[2001:db8:0:1:99a4:6159:198f:b309]
```

-l *timelimit*

Specifies the time limit in seconds for each DAP operation.

-L

Prints entries in LDIF format (-B is implied).

-M

Does not multicast; limits search to a single directory.

-n

Shows what would be done, but does not actually do it. Use with the -v option for debugging.

-p *dap-port*

Specifies the port on directory host computer. If you do not specifiy this, the default of 102, the OSI port, is used.

You can combine the -H and -p arguments into a single argument. For example, you can replace the options on the first line with those on the second:

```
-H [2001:db8:0:1:99a4:6159:198f:b309] -p 19389
-H [2001:db8:0:1:99a4:6159:198f:b309]:19389
```

-s { base | one | sub}

Specifies search scope.

-t *dir*

Writes values to files in the specified directory.

-T

Times the search (no search results printed).

-v

Runs in verbose mode.

-w *password*

Specifies the bind password, which is used for simple authentication.

-z *number-entries*

Specifies the size limit (in entries) for search.

-Z [*ssld_config_filename*]

Specifies that the tool should start a TLS request.

Use -ZZ to require a successful response from the DSA.

ssld_config_filename

Specifies the name of the configuration file used by the -Z option. The default file is:

`DXHOME/config/ssld/dxldap.conf`

This configuration file is a text file. It must contain a line starting TLS_CACERT, and can optionally contain a second line starting TLS_REQCERT. Its format is as follows:

`TLS_CACERT` *`trusted_pem_file`*
`[TLS_REQCERT {allow|demand|hard|never|try}]`

In the configuration file, the lines have the following meaning:

TLS_CACERT *trusted_pem_file*

Specifies the file that contains certificates for all of the Certificate Authorities the client will recognize.

trusted_pem_file

Species the trusted pem file.

This must be an absolute reference to a full path, without environment variables.

Do not enclose trusted_pem_file in quotes.

[TLS_REQCERT {allow|demand|hard|never|try}]

Specifies what checks to perform on server certificates in a TLS session, if any.

If this line is missing, the system uses TLS_REQCERT demand

The keywords have the following meaning:

- allow - The client will request a server certificate and if no certificate is provided, the session proceeds normally. If a bad certificate is provided, it will be ignored and the session proceeds normally.
- demand - The client will request a server certificate and if no certificate is provided, or a bad certificate is provided, the session is immediately terminated. This is the default setting.
- hard - This is a synonym for *demand.*
- never - The client will not request or check any server certificate.
- try - The client will request a server certificate and if no certificate is provided, the session proceeds normally. However, if a bad certificate is provided, the session immediately terminates.

Example: dxldap.conf file on a Windows System, Specifying the TLS_REQCERT Setting

```
TLS_CACERT  c:\program files\CA\Directory\dxserver\config\ssld\trusted.pem
TLS_REQCERT allow
```

Example: dxldap.conf file on a UNIX System, Using the Default TLS_REQECRT Setting

```
TLS_CACERT  /opt/CA/Directory/dxserver/config/ssld/trusted.pem
```

filter

An RFC2254-compliant LDAP search filter.

attributelist

Specifies a space-separated list of attributes to retrieve. If no attribute list is given, all attributes are retrieved.

Example: Search and Results

This example uses the Democorp sample directory supplied with CA Directory. You can repeat this example as a training exercise.

Use the following command to search:

```
%dxsearch -L -h 192.168.19.202:19389 "(sn=horsfall)"
```

The results appear like this:

```
dn: cn=Murray HORSFALL,ou=Repair,ou=Operations,o=Democorp,c=AU
oc: organizationalPerson
oc: newPilotPerson
oc: quipuObject
cn: Murray HORSFALL
sn: HORSFALL
title: Information Technology Manager
telephone: 797 8877
description: Replacements
mail: Murray.HORSFALL@Democorp.com
postalAddress: 173 Toorak Pde $ Berkeley NSW
postalCode: 2506
```

If you send the output to an LDIF file, you can edit the file contents and use the DXmodify tool to implement the changes.

```
dxsearch -L -h yourhost:19389 "(sn=horsfall)" > h-modify.ldi
```

DXsyntax Tool—Check the Configuration of DSAs

Use the DXsyntax tool to check the configuration of one or more DSAs. The tool is most easily run on the server hosting those DSAs, but it can be run remotely, provided that the machine running it has access to the configuration directory of the server hosting the DSAs.

This command has the following format:

`dxsyntax [dsaname]`

dsaname

Specifies the name of the DSA. This can be a file name pattern.

Example: Check all DSAs

To check the configuration of every DSA, run the following command:

```
dxsyntax
```

Example: Check all DSAs Starting with "rk"

To check all DSAs whose names start with *rk*, run the following command:

`dxsyntax rk*.`

How the DXsyntax Tool Works

The utility runs through the configuration files of each DSA in turn, reporting a detailed error message if it finds a problem. The error message specifies the line and file where the error was detected (the actual error may be earlier), and provides details on the error condition. Only one error is reported for each DSA checked, because a single error can cause a cascade of problems that is overwhelming.

If no errors are found, DXsyntax exits without a message, and with a return code of 0. This is handy for use in routine test scripts. If one or more errors are found, DXsyntax exits with an error message and a non-zero return code.

Note: The DXsyntax relies on the DXHOME environment variable. DXHOME must be set to the home path of DXserver. This is done automatically when CA Directory is installed. The DXsyntax tool expects the DSA configuration files to be located in the config folder under the path in DXHOME.

ldif2dxc Tool—Convert Schema from LDIF to CA Directory Format

Use the ldif2dxc tool to convert LDAP schema in LDIF format into the CA Directory schema configuration format (.dxc). The tool can also update an existing schema.txt file.

This command has the following format:

`ldif2dxc [options] [outfile]`

options

Denotes one or more of the following options:

-b *badfile*

Write bad schema records to the specified file.

-f *file*

Read input from the specified file.

-m *map*

Get OIDs from the specified map file.

-M *oid_arc*

Generate missing OIDs from 'oid_arc'. For example x.y.z. generates x.y.z.0, x.y.z.1, etc. x.y.z.34 generates x.y.z.34, x.y.z.35, etc.

-x *dxcFile*

Exclude schema that is defined in the specified .dxc file.

-Z *schema*

Append new schema definitions in DXtools schema file format to the specified file.

outfile

Specifies the name of the file into which the LDIF data will be saved.

-H (or any non-existent option)

Displays usage and option information

Example: Convert the Unique Part of the Schema

You want to convert the schema in the new_schema.ldif file. In this case, you are not interested in the entire external schema, but rather schema unique to the external source. The local directory also typically sources a single DSA schema group file (for example, dxmanager.dxg).

To convert only the schema unique to the external source, you want to instruct the tool to exclude an existing schema file. In addition, you want to instruct the tool not to redefine any internal DSA schema definitions.

Enter the following command:

```
ldif2dxc -f new_schema.ldif -x dxmanager.dxg new_schema.dxc
```

Example: Convert with Errors

While converting the schema in the new_schema.ldif file, the tool stops with an error because of schema incompatibility with CA Directory and does not produce any output. The problem may be caused by an unsupported syntax or matching rule, or an error in the published schema. In these cases, you can instruct the tool to write any bad schema to a bad file but continue writing the good schema to your output file. You can inspect the bad file and decide whether it is worthwhile to correct any of the errors (for example, remove an obscure matching rule for substrings), and then rerun the tool until you have what you need.

To run the tool with a bad file, enter the following command:

```
ldif2dxc -b bad.txt -f new_schema.ldif -x dxmanager.dxg new_schema.dxc
```

Example: Convert with a Map File

Some LDAP directories publish OIDs as labels rather than dotted decimal strings (for example, *xyConfig-oid*). These object classes and attributes do not load into the directory. Instead, the tool assigns them temporary OIDs off the CA Directory arc to enable you to load the new schema, but this solution may not be suitable for all directory implementations.

If the dotted decimal form of these object class and attribute OIDs are available, you can create a map file, and instruct the tool to look up these label OIDs in the file and substitute the labels by the dotted decimal OIDs.

The map file is a three-column comma-separated value (CSV) file with the following format, for example:

```
-------------------------------------
#
# format: objectClass, attributeType, oid
#
# objectClasses
xyConfig-oid,,1.2.3.4
xyAdmin-oid,,1.2.3.5
# attributeTypes
,abstract-oid,1.2.4.5
,aci-oid,1.2.4.6
-------------------------------------
```

You map a dotted decimal OID to either an object class or an attribute type, but not both.

To run the tool with a map file, enter the following command:

```
ldif2dxc -b bad.txt -f new_schema.ldif -m map.txt -x dxmanager.dxg new_schema.dxc
```

Example: Convert with Label OIDs

If there are a large number of "label" OIDs (e.g. xyConfig-oid) in the LDIF schema file it may take a long time to add an entry for everyone in a map file. An alternative is to specify an OID arc that the tool will use in place of any label OID, incrementing it for each label OID it finds.

If your arc is new, you can specify it with a trailing '.', and the tool will start incrementing it from 0. If some OIDs have already been assigned against this OID arc, you can specify the next available OID, and the tool will start incrementing from that one.

To replace the map file in Example 15 with a new OID arc of "1.22.333.444.", on a UNIX system, enter:

```
ldif2dxc -b bad.txt -f new_schema.ldif -M 1.22.333.444. -x dxmanager.dxg -Z
$DXHOME/bin/schema.txt new_schema.dxc
```

If the tool encountered the OID label xyConfig-oid in new_schema.ldif, it will assign it an OID of 1.22.333.444.0. If it then encountered the OID label abConfig-oid, it will assign it an OID of 1.22.333.444.1, etc.

If twenty OIDs from this arc were assigned, the next available OID would be 1.22.333.444.20. If you were to perform another integration with schema from new_schema2.ldif, to avoid clashing with existing OIDs in the directory, on a UNIX system, enter:

```
ldif2dxc -b bad.txt -f new_schema2.ldif -M 1.22.333.444.20 -x dxmanager.dxg -Z
$DXHOME/bin/schema.txt new_schema2.dxc
```

If the tool encountered the OID label cdConfig-oid in new_schema2.ldif, it will assign it an OID of 1.22.333.444.20. If it then encountered the OID label efConfig-oid, it will assign it an OID of 1.22.333.444.21, etc.

Example: Convert an OpenLDAP SLAMD Schema

The OpenLDAP SLAMD is a non-standard schema representation

However, ldif2dxc is tolerant of other such schema types:

```
ldif2dxc -f slamd.openldap.conf -x x500.dxc slamd.openldap.conf.dxc
```

ldifdelta Tool—Calculate the Difference Between LDIF Files

Use the ldifdelta tool to calculate the change, or delta, between two LDIF files. The ldifdelta program is an offline directory synchronization tool based on the LDAP directory interchange format. You can use ldifdelta to fully or partially synchronize directories.

You must do two things before using the ldifdelta tool:

1. Get the two LDIF files you want to compare, *oldfile* and *newfile*, say.

   To do this, use the DXsearch tool (with the -L option) or the DXdumpdb tool.

2. Sort the LDIF files using the ldifsort tool.

ldifdelta can produce an output file, which, is a file containing LDIF change records. You can use the DXmodify tool to apply these LDIF change records to the sorted *oldfile*, and so update it to *newfile*.

The ldifdelta tool has the following limitations:

- It compares distinguished name in a case insensitive way, not by schema matching rules.
- When comparing URL values, ldifdelta compares only the file names.

  It does not attempt to interpret the nature or contents of the file that the URL references.
- It can only be used to delta LDIF data files.

  You cannot compare LDIF change files.

The ldifdelta tool ignores the following operational attributes:

- Arc 2.5.12, "X.500 schema":
  - dsaType
- Arc 2.5.18, "X.500 schema":
  - createTimestamp
  - modifyTimestamp
  - creatorsName
  - modifiersName
  - subschemaSubentry
- Arc 1.3.6.1.4.1.3327.6, "DXserver schema":
  - dxUpdatedByDisp
  - dxEntryCount
  - dxTotalEntryCount

- dxProxyRole
- dxProxyUser
- dxDynamicAccess
- dxDeleteTimestamp
- dxServerVersion
- dxPwdLoginTime
- dxPwdLastChange
- dxPwdHistory
- dxPwdFailedAttempts
- dxPwdFailedTime
- dxPwdLocked
- dxPwdGraceLogins
- dxPwdIgnoreExpired
- dxPwdMustChange
- dxPwdIgnoreSuspended
- dxPwdGraceUseTime
- dxErrorReason
- dxAttrOverlayReferenceSubordinate
- dxAttrRdnValue
- dxNewSuperior

This command has the following format:

`ldifdelta [-x] [-v] -S` *`dsaname oldfile newfile`* `[`*`outfile`*`]`

-x

Ignores X.500 and DXserver operational attributes.

-v

Runs in verbose mode.

-S *dsaname*

Specifies the DSA server containing schema definitions.

Note: The DSA name is only used for schema checking. This does not imply that the LDIF and DSA name used are linked in any way.

oldfile

Specifies the outdated file to produce the delta for.

newfile

Specifies the more recent file to compare against *oldfile*.

outfile

(Optional) Specifies the output file containing the differences between *newfile* and *oldfile*.

If you do not specify this file, ldifdelta produces its output to the standard output.

Example: Using ldifdelta and ldifsort Together

This example makes the old directory the same as the reference directory:

```
dxsearch -L -h oldhost "(oc=*)" > old.ldif
dxsearch -L -h referencehost "(oc=*)" > ref.ldif
ldifsort old.ldif old_sorted.ldif
ldifsort ref.ldif ref_sorted.ldif
ldifdelta -x -S DSA1 old_sorted.ldif ref_sorted.ldif | dxmodify -h oldhost
```

ldifsort Tool—Sort LDIF Records

Use the ldifsort tool to sort an LDIF file or input stream for the given attribute type.

The tool can sort the LDIF record based on any attribute. By default, the ldifsort tool sorts LDIF records based on their distinguished names. This ensures that each record is followed by its immediate subordinates.

You may opt to sort in descending order. You can use this option, for example, when deleting the entries in an LDIF file from the directory. This sorts the LDIF file on DN in descending order, thus ordering subordinates before superior entries. The LDIF file can then be passed to dxmodify to delete these entries.

Although the default sort is by DN, you may wish to use another attribute, for example, to sort employee records in an LDIF file based on their employee numbers for administration, or based on telephone numbers to allocate spare lines.

The ldifsort tool can cope with bad input records. See the -b option. A record is bad for one of the following reasons:

- Problem comparing DNs
- Multiple DNs found in a single entry
- Invalid LDIF format
- Problem decoding base64 value
- Sort-by attribute not found
- Problem normalizing the DN
- Duplicate Entry (unless -U is specified. See -U and -u option).

This command has the following format:

`ldifsort [options] infile [outfile]`

options

Denotes one or more of the following options:

-a *attr*

Sorts entries based on the specified attribute. The default sort attribute is dn.

-b *file*

Writes bad input records to the specified file. Each bad input record is accompanied by the reason it is considered bad.

If -b is not specified, bad records are silently discarded, except that the final summary report gives the count of bad input records.

-d

Sorts in descending order. The default is to sort in ascending order.

-m *count*

Specifies the number of records that are put into each sorting bucket. The default is 200. For the fastest sort time, set this option to the square root of the number of entries in the file (-m count = √ number of entries in file).

-r *block*

Specifies the number of sorting buckets to allocate at a time. The default is 10,000.

-s *bytes*

Specifies the size of read buffer for each bucket. The default is 2,048 bytes.

-t *dir*

Specifies the directory to use for temporary files.

-u

(Default). Checks for duplicates. Duplicate records are considered bad input records.

-U

Does not check for duplicates, so duplicates are considered good input records.

-v

Runs in verbose mode. In this mode, diagnostics are sent to standard error.

infile

Specifies the file to be sorted.

outfile

Specifies the file to which output is to be written. The default output is standard out.

Example: Make Directories the Same

This example makes the old directory the same as the reference directory:

```
dxsearch -L -h oldhost "(oc=*)" > old.ldif
dxsearch -L -h referencehost "(oc=*)" > ref.ldif
ldifsort old.ldif old_sorted.ldif
ldifsort ref.ldif ref_sorted.ldif
ldifdelta old_sorted.ldif ref_sorted.ldif | dxmodify -h oldhost
```

Reasons for BAD Records

The most common reason for the ldifsort tool created a BAD record is a duplicate entry.

A BAD record is written to the BAD file, if one is specified (-b option), alongside the reason it is considered BAD, which can be one of the following:

- Duplicate Entry
- Problem comparing DNs
- Multiple DNs found in a single entry
- Invalid LDIF format
- Problem decoding base64 value
- Sort-by attribute not found
- Problem normalizing the DN

Chapter 5: System Messages

For assistance, contact CA Support at http://ca.com/support.

This section contains the following topics:

DXserver Alarm Messages

This section lists the messages that can appear in an alarm log file.

DSA_E1000 DSA has invalid authenticationLevel

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1010 Bad tag: itemOrUserFirst

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1060 Illegal specificationFilter

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1070 ProtectedItems: allAttributeValues not allowed

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1080 ProtectedItems: attributeValue not allowed

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1090 ProtectedItems: selfValue not allowed

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1100 ProtectedItems: No protected items

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1170 Unable to open pid file: '$1'

Reason:

On startup, the DSA could not create a pid file under DXHOME/pid.

Action:

Ensure that the DSA user has write access to this directory.

DSA_E1180 Unable to write to pid file: '$1'

Reason:

On startup, the DSA could not write to the pid file under DXHOME/pid.

Action:

Ensure that the DSA user has write access to this file.

DSA_E1250 DSA stopping but multiwrite queues exist

Reason:

The DSA was forced to shut down while holding multiwrite queues of updates for peers.

Action:

You need to manually resynchronize the multiwrite peer DSAs. For more information on how resynchronize a multiwrite DSA, see the *Administration Guide* chapter on Replication.

DSA_E1260 DSA stopping but operations in progress. If running multiwrite then data may be out of sync

Reason:

The DSA was forced to shutdown while operations were in progress.

Action:

You might need to manually resynchronize the multiwrite peer DSAs.

DSA_E1280 Error in initialization files

Reason:

On startup, the DSA has detected a problem in its configuration files.

Action:

The DSA trace log indicates the issue that needs to be resolved before the DSA can be started.

DSA_E1290 No stack and no console

Reason:

The communications software has not been initialized at startup.

Action:

Ensure that the host and the DSA are properly configured.

DSA_E1300 Failed to initialize licensing

Reason:

On startup, the DSA failed to load the licensing library lic98, and so cannot ensure that it is correctly licensed.

Action:

Contact CA support for assistance to ensure that lic98 is installed and functioning correctly.

DSA_E1310 Cannot open local console

Reason:

On startup, the DSA console failed to initialize.

Action:

Ensure that the port configured for the DSA console is not currently in use.

DSA_E1380 Cache is full - index limit reached ($1)

Reason:

The DSA index cannot hold any more values.

Action:

Use the set max-cache-index-size command to increase the cache index size to a value larger than the required number of attributes, then restart the DSA.

DSA_E1490 DSA cache disabled

Reason:

The DSA cache is no longer running. The most common reasons are:

- Machine resource limits have been reached
- The maximum cache size was exceeded.

Action:

See the trace log to see why the cache was disabled.

DSA_E1870 DSA cache has been restarted so all cache data was lost.

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E1960 Detected XML schema version '$1' but only support version '$2'

Reason:

The DSA has detected an XML configuration file that is too new. The XML schema defines the information held by DXmanager.

Action:

Ensure that DXmanager has the same version of the XML schema as the DSAs that it manages.

DSA_E1990 Cannot register SNMP address

Reason:

The DSA is trying to register a port as the SNMP port to listen on, but the port is already being used.

Action:

Ensure that the configured SNMP port is not used.

DSA_E2220 Cannot register address

Reason:

On startup, the DSA cannot listen on the interface configured.

Action:

Ensure that the interface address or hostname, and the port are valid and not currently in use.

DSA_E2230 Too many registrations

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E2240 DSA has multiple interfaces that resolve to the same address

Reason:

The DSA configuration specifies multiple addresses for a DSA, but two or more of these address are identical or resolve to identical physical addresses.

Action:

Ensure multiple address specifications resolve to multiple addresses.

DSA_E2340 Out of memory

Reason:

The DSA has run out of memory

Action:

Reduce the number of results returned by single searches by breaking them into a number of separate, smaller searches, each returning part of the result. Also, consider increasing the amount of virtual memory.

DSA_E2760 Multiwrite: Operation disabled for DSA $1

Reason:

The multiwrite queue for the DSA has been purged.

Action:

You need to manually resynchronize the DSA.

DSA_E2780 Multiwrite: request not queued - $1 PDU parse failed

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_E2860 Too many NSAPs

Reason:

A DSA can only listen on a maximum of ten interfaces.

Action:

None.

DSA_E2910 Cannot configure self

Reason:

The DSA cannot use its prefix.

Action:

Ensure that the prefix for the DSA is correct and is supported by the schema.

DSA_E2920 Cannot have no-server-credentials with multi-write-disp-recovery

Reason:

Multi-write-disp-recovery does not work if the DSA has no-server-credentials.

Action:

Change the DSA configuration, so that it it does not have no-server-credentials.

DSA_E2940 Attempt to set multi-write-disp-recovery on a router or relay DSA

Reason:

Multi-write-disp-recovery has no meaning for a router or relay DSA.

Action

Do not try to set multi-write-disp-recovery on a router or relay DSA.

DSA_E2980 Self not defined in knowledge files or XML

Reason:

A DSA server file (.dxi) does not source a knowledge definition that describes the DSA being started.

In a text file based configuration, the .dxi file name does not match the knowledge file given by 'set dsa name' and any source knowledge file/group.

In a DXmanager based configuration, a server file might have been updated without using DXmanager. Doing this is not recommended.

Action:

If you are using DXmanager, then ensure you use DXmanager to modify the server files.

If you are using knowledge files, ensure the DSA name of the .dxi file matches the label used in the set dsa command.

DSA_E3000 Invalid partition definition, expected '$1'

Reason:

The definition of the horizontal partition is not valid. The definition is missing one or square brackets or angles brackets (chevrons).

Action:

A horizontal partition definition in a DSA prefix must be of the following form:

attr **"[*hashn* (*total*)=*id*]"**

attr

Defines the naming attribute that is being partitioned.

hashn

Specifies the hash, and is one of the following:

- hash1
- hash2

total

Defines the number of partitions.

id

Specifies one partition, and is a number from zero to number of partitions - 1.

DSA_E3010 Invalid partition criteria specified

Reason:

The definition of the horizontal partition specifies an invalid hash.

Action

Ensure the hash definition is valid.

DSA_E3020 Partition count must be positive

Reason:

The definition of the horizontal partition specifies a non-positive partition count.

Action

Ensure the partition count is positive.

DSA_E3030 Partition criteria must be the same for each partition DSA $1

Reason:

Not all DSAs in a horizontal partition set have the same hash definition.

Action

Ensure all DSAs in a horizontal partition set have the same hash definition.

DSA_E3040 Partition count must be positive $1

Reason:

The definition of the horizontal partition specifies a non-positive partition count.

Action

Ensure the partition count is positive.

DSA_I1020 UserClasses: ignoring UniqueIdentifier

Reason:

When the DSA applies ACLs on names using the NameAndOptionalUID syntax, it ignores the UID part of the name.

Action:

None.

DSA_I1030 acf_copyUserClasses: duplicate group

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_I1220 DSA started: $1

Reason:

The DSA has started successfully.

Action:

None.

DSA_I1230 DSA reloading configuration

Reason:

The DSA is reloading its configuration.

Action:

None.

DSA_I1240 DSA shutting down

Reason:

The DSA is shutting down.

Action:

None.

DSA_I1340 DSA has been forced to shutdown

Reason:

(Windows) The DSA was shut down with the forcestop option. All updates were successfully replicated.

Action:

None.

DSA_I1350 DSA shutting down as queues have flushed

Reason:

The DSA has flushed its multiwrite queues and is now safely shutting down.

Action:

None.

DSA_I1360 DSA received command to stop

Reason:

The DSA has received a command to stop.

Action:

None.

DSA_I1880 Loading XML version $1

Reason:

This shows the version of XML that the DSA is using.

Action:

None.

DSA_I2010 SNMP: Requested community not supported: $1

Reason:

The DSA is using a snmp-poll-community string, and the SNMP client is polling using an incorrect community.

Action:

Ensure that the SNMP client uses the community string configured for the DSA. Use the commands *set snmp-poll-community* and *set snmp-trap-community*.

DSA_I2200 Mutex $1 from $2 held too long

Reason:

A mutual exclusion lock (mutex) should only be held briefly by any one process. If a mutex is held too long, then DXserver performance can suffer.

Action:

Contact CA support if you notice performance degradation.

DSA_I2210 Thread $1 no longer running

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_I2450 Accepted console request from $1

Reason:

This is an audit trail of a user's connection to the DSA console.

Action:

None.

DSA_I2490 Accepted console request for user $1 from $2

Reason:

This is an audit trail of a user's connection to the DSA console.

Action:

None.

DSA_I2690 Multiwrite-DISP: Update completed: $1

Action:

The peer DSA is now synchronized.

Action:

None.

DSA_I2740 Multiwrite-DISP: Attempting to update peer: $1

Reason:

The DSA is in the process of updating a peer.

Action:

None.

DSA_I2830 Clearing multiwrite queue: $1

Reason:

This is a notification that a user has issued the command clear multi-write-queue.

Action:

If you issue the clear multi-write-queue command and want to synchronize multiwrite DSAs, you need to do the synchronization manually.

DSA_I3200 License:

Reason:

When a DXserver is started, a licensing message appears that displays the DXserver version, platform, 32/64 bit, prefix and entry count (if applicable). This can be used by auditing applications to determine if the current deployment is correctly licensed.

Action:

None

DSA_I2850 Aborting all non-peer connections during recovery from peer

Reason:

A DSA running no service while recovering has received notification from a peer that it is not synchronized. It will drop all non-peer (client) connections until it is synchronized.

This is fairly rare, and occurs because the network connections are repeatedly breaking and reforming.

Action:

If this keeps occurring, then you should investigate the network connections.

DSA_W1160 DSA shutdown forced

Reason:

(Windows) The DSA was shut down by the command dxserver forcestop.

Action:

None.

DSA_W1270 Loading configuration while processing queues

Reason:

When DXserver receives an 'init' it will wait for updates in progress to complete before reloading configuration. The updates have taken more than 30 seconds so the configuration will be reloaded

Action:

This is not a problem in itself, but might be a symptom of something else, such as an intermittent hardware problem.

DSA_W1430 DSA cache index almost exhausted ($1)

Reason:

The DSA index is growing and has almost the maximum number of values it can hold.

Action:

Consider increasing max-cache-index-size to a value larger than the required number of attributes. You need to restart the DSA after changing max-cache-index-size for the new setting to take effect.

DSA_W1450 Cache is $1% full

Reason:

This indicates how full the cache is, as a percentage of the value of max-cache-size.

Action:

If the cache is nearing 100%, then consider increasing max-cache-size. You need to restart the DSA after changing max-cache-size for the new setting to take effect.

DSA_W1830 LDAP Abandon: Invalid invoke ID

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W1840 LDAP Abandon: Cannot abandon READ

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W1890 DSA $1 has 'no-server-credentials' set

Reason:

A DSA that is not LDAP has the setting no-server-credentials on.

Action:

Use DXmanager to disable no-server-credentials on the DSA.

DSA_W1900 DXconsole connection alert now off. Successful DXconsole connections will no longer produce any alerts.

Reason:

Normally, you turn the DSA console connection alerts on for auditing requirements. This message alerts the user that somebody has turned these off, which may be a breach of security.

Action:

Use the command set dxconsole-connect-alert to turn DSA console alerts off or on.

DSA_W1910 check-roles-with-dsa-credentials is a deprecated command. Use the setting trust-dsa-triggered-operations instead

Reason:

Deprecated command was used.

Action:

Use the setting trust-dsa-triggered-operations.

DSA_W1920 check-unique-attrs-with-dsa-credentials is a deprecated command. Use the setting 'trust-dsa-triggered-operations' instead.

Reason:

Deprecated command was used.

Action:

Use the setting 'trust-dsa-triggered-operations'.

DSA_W1940 Setting trace level to 'asn'

Reason:

Trace level has been set to asn. Normally this trace level is only used for debugging purposes.

Action:

Be aware that using the trace level 'asn' can cause password information to be written to the DSA log files.

DSA_W1970 Old-style configuration: changing item '$1'

Reason:

Configuration has multiple occurrences of the same configuration item.

Action:

To avoid ambiguity, remove duplicates.

DSA_W2000 Cannot initialise SNMP trap - no memory

Reason:

The DSA has run out of memory

Action:

Reduce the number of results returned by single searches by breaking them into a number of separate, smaller searches, each returning part of the result. Also, consider increasing the amount of virtual memory.

DSA_W2020 Cannot create hash table - no memory

Reason:

The DSA has run out of memory

Action:

Reduce the number of results returned by single searches by breaking them into a number of separate, smaller searches, each returning part of the result. Also, consider increasing the amount of virtual memory.

DSA_W2040 Cannot send SNMP response to $1

Reason:

The DSA cannot contact the SNMP client requesting information.

Action:

Ensure the network is properly configured and that the SNMP client is available.

DSA_W2330 Accept failed: $1

Reason:

The DSA failed to accept a new call. The message includes the reason for the the failure.

Action:

Act according to the reason given in the message.

DSA_W2430 Syntax not found for $1

Reason:

This message is only produced during a DSA init. Because the init procedure clears the schema, an operation in progress suddenly has no valid syntax.

Action:

Retry the operation once the init has completed.

DSA_W2440 Rejected console request from $1

Reason:

The DSA console rejected a connection request because it already has a connection.

Action:

The DSA console supports only one connection at a time.

DSA_W2460 Invalid credentials for user $1 from $2

Reason:

Authentication failed for the supplied user DN and password.

Action:

Ensure the user is a configured DSA console user and the password is correct.

If these messages appear frequently, this may indicate an attack from a malicious user trying to gain DSA console access.

DSA_W2480 Invalid console password entered from $1

Reason:

The password supplied does not match the configured DSA console password.

Action:

Ensure password is correct. If these appear frequently may indicate an attack from a malicious user trying to gain DSA console access.

DSA_W2530 DISP bind failure: DISP agreement authentication level is not supported by DISP responder

Reason:

The DISP agreement uses an authentication level that does exist on the link between the sender and responder.

Action:

Ensure that the DISP initiator and responder are using the same authentication level.

DSA_W2540 DISP Update unsuccessful

Reason:

The DISP initiator failed to send a DISP update to the responder.

Action:

Ensure that the DISP is properly configured and that the DISP initiator can contact the DISP responder.

DSA_W2550 Duplicate shadow DSE

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W2560 Attribute not found

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W2590 Cannot set update time

Reason:

For each peer DSA, the DISP DSA records the time when it has successfully updated the peer DSA.

If the DSA cannot set the update time, then it will resend a DISP update.

Action:

If this occurs repeatedly, then contact CA support. One symptom of this problem is that the DSA repeatedly sends the same updates to the peer, but the updates fail to get applied.

DSA_W2600 Cannot set update time dsa name

Reason:

For each peer DSA, the DISP DSA records the time when it has successfully updated the peer DSA.

If the DSA cannot set the update time, then it will resend the DISP update.

Action:

If this occurs repeatedly, then contact CA support. One symptom of this problem is that the DSA resends updates to the peer, but the updates fail to get applied.

DSA_W2620 Cannot get last update time

Reason:

The DSA failed to retrieve the last time it updated a peer DSA. It will periodically retry to get the last update time.

Action:

If this continues, contact CA support.

DSA_W2650 Cannot get multiwrite DISP last update time

Reason:

The DSA failed to retrieve the last time it updated a peer DSA. It will periodically retry to get the last update time.

Action:

If this continues, contact CA support.

DSA_W2660 Invalid dynamic role definition: $1

Reason:

The configured dynamic role contains an invalid LDAP URL.

Action:

The LDAP URL specifies the location of role members. It should be of the form: ldap:///<filter>, as specified in RFC 2255.

DSA_W2680 Multiwrite queue ($1) greater than $2% full

Reason:

The multi-write queue for the DSA specified in the message is filling up.

Action:

Ensure that the multi-write peer DSA is running and is reachable.

DSA_W2710 Multiwrite-DISP: Failed attempt to update peer: $1

Reason:

The DSA failed to update a peer DSA using multiwrite DISP. This usually occurs when a DSA is not properly configured, for example, there may be authentication or address problems.

Action:

The trace log should provide the reason for the failure.

DSA_W2720 Multiwrite: $1DSA '$2' cannot be contacted

Reason:

The DSA tried to replicate an update, or to flush a multiwrite queue, to a multiwrite peer DSA, but it cannot contact the peer.

Action:

Ensure that the peer DSA is running and reachable.

DSA_W2730 Multiwrite-DISP: Unable to update peer $1 - cannot connect

Reason:

The DSA tried to replicate to a peer using multiwrite DISP, but it cannot contact the peer DSA.

Action:

Ensure that the peer DSA displayed is running and reachable.

DSA_W2770 Multiwrite: Operation suspended for DSA $1

Reason:

The multiwrite queue to the peer DSA in the message has been purged but multiwrite is enabled. The source DSA will use multiwrite DISP (instead of multiwrite) to update the peer DSA.

Action:

Ensure that the peer DSA is running and reachable, so that multiwrite DISP can resynchronize data and multi-write replication can continue normally.

DSA_W2800 Non-empty update for LDAP peer: $1

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W2820 Multiwrite-DISP: Invalid update strategy

Reason:

This message should not occur.

Action:

Contact CA Support.

DSA_W2870 allow-search label '$1' not found

Reason:

The role-based allow search label for the current user does not exist in the DSA's configuration.

Action:

Ensure that the allow-search labels used for the role-based dxAllowSearch attribute exist in the allow-search list.

DSA_W2890 PwdPolicy: Account resetting requires a password policy request control

Reason:

There is no password policy control (an LDAP control)

Action:

Create a password policy control. See LDAP Controls, in Connect to LDAP clients in the *Administration Guide*.

DSA_W2900 Multiple duplicate attribute values found

Reason:

An attempt was made to add a duplicate attribute value but unique-attrs is set.

Action:

None required.

DSA_W2930 Views: phase $1 search for view $2 is marked result-required but did not return a result

Reason:

A phase search marked as result-required did not return any entries.

Action:

Ensure that data used for linking phases exists.

DSA_W2950 Data integrity compromised - internal '$1' failed but reference was updated

Reason:

This can occur when using overlays. An update was attempted across both an overlay and a reference DSA, but only the reference update worked.

Action:

Investigate the reason for the overlay update failure.

Installation Error Messages on Windows

This section lists the messages that can appear when you install CA Directory on Windows.

Error 267 or Error 1606

Reason

These errors occur when there are inconsistencies with the Windows Installer DLL.

This usually occurs because a reboot was not performed when Windows Installer was upgraded. This problem can also occur when you are upgrading from eTrust Directory 3.6.

Action

To prevent this problem from occurring or to fix this error if it has already happened, use the CA Directory Cleanup tool, which is located under Unsupported on the CA Directory installation CD. To run this tool, use the following command:

```
CleanReg.exe -fixup267
```

Error 1605

Could not retrieve Version String

Reason

This error appears when the registry has rogue entries under the Windows Installer areas. These are usually from an aborted or failed uninstallation.

Action

Use the CA Directory Cleanup tool, which is located under Unsupported on the CA Directory installation CD.

To run this tool, use this command:

```
CleanReg.exe -all
```

Error 1639

Invalid Command Line Argument

Reason

This error can occur when you install CA Directory using commands.

Action

If you receive this error, check the following:

- All arguments must be in the form of =. i.e ETRDIR_SILENT_INSTALL=1
- All characters A-Z from the PROPERTY must be uppercase.

All paths specified in the VALUE must have quotes if they contain spaces. For example, FOLDER="c:\Program Files\CA"

Chapter 6: File Structure and File Formats

This section contains the following topics:

File Structure

After installation, the CA Directory files use the following directories:

```
DXHOME
    |__ bin
    |__ config
    |      |__ access
    |      |__ autostart (UNIX only)
    |      |__ knowledge
    |      |__ limits
    |      |__ logging
    |      |__ replication
    |      |__ schema
    |      |__ servers
    |      |__ settings
    |      |__ ssld
    |__ data
    |__ install  (UNIX only)
    |__ logs
    |__ pid
    |__ samples
    |__ uninstall  (UNIX only)
```

bin

Holds all binary executables and batch files for CA Directory.

config

Holds the current configuration files, and some Readme files for your reference. DSAs always start with the configuration information stored in this directory. We recommend that you give read-only permissions to files in these directories to prevent accidental overwriting.

Each subdirectory contains the configuration file(s) that deal with a component of the configuration.

access

Holds configuration files that specify access control rules.

autostart

(UNIX only) Tracks the DSAs that must start at the same time as the operating system.

knowledge

Holds the knowledge files for DSAs if DXmanager is not used.

limits

Holds configuration files that specify administrative limits, such as size limits and time limits.

logging

Holds configuration files that specify trace levels and log file names.

replication

Holds configuration files that specify replication agreements.

schema

Holds configuration files that specify schema definitions.

servers

Holds initialization files that specify startup information. An initialization file must exist for each DXserver on the host computer.

settings

Holds configuration files that specify operational settings and controls.

ssld

Holds .pem certificates for trusted CA and server entities.

data

Holds the data store file (.db) for each DSA.

install

(UNIX, for CA Directory use only) Contains installation files and environment information.

logs

Holds all log output for CA Directory. You can specify another directory.

pid

(CA Directory use only) Contains information about currently running processes. The *pid* directory is empty immediately after an installation, but when the services are run, files are created in that directory.

samples

Contains demonstration utilities and test script files. Each sample has its own subdirectory and an associated Readme file.

uninstall

(UNIX only) Contains uninstallation scripts.

LDIF

LDIF is a format suitable for describing directory information or changes to be made to directory information.

LDIF files are text files that store directory information in LDIF. You can use LDIF files to transfer directory information between LDAP directory servers or to describe a set of changes to be applied to a directory.

LDIF files are useful for these tasks:

- Load new entries into a directory
- Restore entries into a directory after recovery
- Dump (export) data from a directory
- Apply bulk edits to directory data

LDIF is defined in the Internet standard RFC 2849. For more information about LDIF, see the draft RFC available at the IETF home page.

Format of Information in an LDIF File

An LDIF file consists of a series of records, separated by blank lines.

Each record describes a directory entry or a set of changes to a directory entry. These two types of records cannot be mixed in an LDIF file.

Information only

If you want to do any of the following tasks, use an information-only LDIF file:

- Load new entries into a directory
- Restore entries into a directory after recovery

Directives for changes

If you want to edit existing data, use an LDIF file that contains directives.

LDIF File Containing Description of an Entry

- Load new entries into a directory
- Restore entries into a directory after recovery
- Dump (export) data from a directory

The first line of an entry is the distinguished name (dn). The rest of the entry is made up of attribute value pairs, separated by a colon (:) and a space.

Each record is separated with a blank line.

The basic form of an LDIF entry is:

```
dn: distinguished-name
attribute-type : attribute-value
attribute-type : attribute-value...
```

Example: A Simple LDIF File with Two Entries

This example shows two LDIF records. An LDIF file that contains these records could be used to load these two entries into a directory:

```
version: 1
dn: cn=Barbara Jensen, ou=Product Development, dc=airius, dc=com
objectclass: top
objectclass: person
cn: Barbara Jensen
sn: Jensen

dn: cn=Barbara Jensen, ou=Product Development, dc=airius, dc=com
objectclass: top
objectclass: person
cn: Barbara Jensen
sn: Jensen
```

LDIF File Containing Changes to Data

An LDIF file can contain instructions for changes to existing entries in a directory. LDIF lets you specify these changes:

- Add or delete an entire entry
- Add, delete, or modify an attribute
- Add an object class to an entry
- Rename an entry by modifying its RDN
- Move an entry to another location in the DIT

```
changetype: add
changetype: delete
changetype: modify

version: 1
dn: distinguished-name
changetype: modrdn
newrdn: name
[deleteoldrdn: 0|1]
[newsuperior]
```

You cannot rename an entry if that entry has children.

Example: One LDIF Change Directive

This example shows a single entry, for which the title will be replaced:

```
version: 1
dn: cn=Murray HORSFALL, ou=Repair,ou=Operations,o=Democorp,c=AU
changetype: modify
replace: title
title: Chief Information Officer
```

Example: An LDIF file Containing Many Change Directives

This example shows a single entry with a series of changes:

```
version: 1
dn: cn=Murray HORSFALL, ou=Repair,ou=Operations,o=Democorp,c=AU
changetype: modify
replace: title
title: Chief Information Officer
-
add: telephone
telephone: 797 8888
-
delete: description
-
replace: postalAddress
postalAddress: 173 Toorak Rd $ South Yarra
postalCode: 3066

dn: cn=Murray HORSFALL, ou=Repair,ou=Operations,o=Democorp,c=AU
changetype: add
title: Chief Information Officer
telephone: 797 8888
postalAddress: 173 Toorak Rd $ South Yarra
postalCode: 3066
```

Encryption, Special Characters, and Binary Data in LDIF

Modern LDIF files all have a version number at the top of the file, which is almost always *version: 1*. Some older LDIF files do not have this, but they follow the same general rules.

Non-ASCII Characters in LDIF Files

An LDIF file uses Unicode strings to store non-ASCII characters, such as Kanji.

Unicode strings are encoded in base-64 form, which is not human-readable. The base-64 data is converted back into the original data when the LDIF file is read into a program, such as the DXloaddb tool.

Example: An Organizational Unit Entry in Japanese:

```
dn:: b3U95Za25qWt6YOoLG89QWlyaXVz
objectclass: top
objectclass: organizationalUnit
ou:: 5Za25qWt6YOo
```

Binary Data

An LDIF file stores the binary data that is stored in the entry. For example, an entry might include a JPEG photo or a certificate.

LDIF Template Files (LDT)

The information contained in a data file is a list of values. To give meanings to these values, you must specify what the values in each field represent. To do this, define an LDIF template (LDT) file.

The LDT files describe the objects contained in the directory. The LDT file follows the LDIF file format, which is used to add directory information to the data values. An LDIF file consists of a series of records separated by blank lines. A record consists of a sequence of lines describing a directory entry. Special substitution tokens are used to place fields from the CSV file into the LDT.

Each line in the LDT file defines a rule that links a field in a CSV data file to a directory attribute or name with an object description. These rules let the tools translate CSV file information into LDIF file formats.

The Format of LDT files

Each record in an LDT file specifies the format of entries at that level. Each record contains a DN and one or more attribute-value template lines.

use the $ character to denote a column in the source csv file. For example $2 gets replaced with the contents of the entry in second column in the csv file. If the escape character (\) precedes the $ character the the $ symbol is interpreted literally. Use \\ to represent the \ character.

Using substitution tokens in the DN line lets you specify leaf nodes and their parents. You can therefore infer hierarchy from the original CSV data. You must explicitly code parent objects in the template file, and they must appear in increasing-depth order in the file before any definition of leaf objects.

To specify a literal number following immediately after a substitution token, use the three-digit form of the token as in the following example:

```
...

Description: $00199 \$ $2 \$ $3
...
```

When this LDT file is interpreted, the csv2ldif program looks at the first three digits after the $ for the column number. In this example, it interprets $00199 to mean find the $001 column of the CSV to substitute, that is column one, and append the characters "99" to it. If you use $199 it substitutes the value from the 199th column of the CSV.

Example: LDIF Template File

```
# Acme template file - LDIF format
# telephonenumber is a multivalued attribute

dn:ou=$4, o="Acme", c=US
oc:organizationalUnit

dn:cn=$1 $2, ou=$4, o="Acme", c=US
oc:organizationalPerson
surname:$2
title:$3
telephonenumber:+61 3 $5
telephonenumber:+61 3 $6
```

CSV

A source data file is a comma-separated list of values. Each source data (CSV) file can contain many lines, and each line in the file should contain information about a single object or entry.

While the default separator is a comma, the DXtools support the use of alternative and user-defined separators.

You should enclose embedded commas within quotes. For example, if Tom Smith's address is contained as a single field, you can include embedded commas in the single address field:

```
Tom,Smith,"36 High Street,Boystown,NY"
```

To accommodate embedded quotation marks, as when applied to a property name in an address field, always use two sequential quotation marks to represent a single quotation mark in text. For example, if Tom Smith used his property name "The Grange" in his address:

```
Tom,Smith,"""The Grange"",High Street,Boystown,NY"
```

Example: CSV Data File

```
Fred,Jones,Manager,Administration,987 6254
Tom,Smith,Sales Person,Sales,987 6251
Bill,Smith,Foreman,Manufacturing,987 6257
Ken,Anderson,Account Manager,Sales,987 6255
Wendy,Murphy,Clerk,Administration,987 6233
Ann,Thompson,Receptionist,Administration,987 6222
Ian,Hall,Boilermaker,Manufacturing,987 6510
Linda,Bates,Accountant,Administration,987 6511
Sam,Campbell,Welder,Manufacturing,987,6510
```

Chapter 7: Limits

This section contains the following topics:

Limits That You Can Change

Limit the Number of Index Entries

Use the set max-cache-index-size Command (see page 82) to limit the number of entries in the cache index. The default is an unlimited number of entries.

Service Limits

You can set limits on the searches that a DSA can perform.

Note: If the DSA tests a user request against any search profile, then it will ignore the *limit list* and *limit search* configuration items for that request. For more information, see Using Search Profiles.

Description of the Limit	How to Set the Limit	Default If Limit Is Not Changed
Prevent a DSA from accepting list requests Prevents a superior DSA from sending list requests to this DSA.	Specify the *Limit list* configuration item in DXmanager	List requests are not prevented
Prevent a DSA from accepting lengthy search requests. Prevents another (superior) DSA from sending lengthy search requests to this DSA	Specify the *Limit search* configuration item in DXmanager	Search requests are not limited
Limit the DSA to only exact searches	Specify the *Limit search exact* configuration item in DXmanager	The DSA is not limited to only exact searches
Prevent all DSAs that service a namespace from accepting update requests Prevents a router DSA from sending update requests to these DSAs	Set the *Read Only* configuration item in DXmanager	The DSAs accept update requests

Operations Limits

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Number of entries that a search or list can return	The service limit set by the client, or if this is not given, the role-based administrative limit *dxSizeLimit*, or if this is not set, the set max-op-size Command (see page 83)	200
Time that a non-update operation can continue to run	The service limit set by the client, or if this is not given, the role-based administrative limit *dxTimeLimit*, or if this is not set, the set max-op-time Command (see page 83)	600
Number of concurrent local operations	set max-local-ops Command (see page 83)	100
Number of concurrent operations per user	set credits Command (see page 64)	5

Bindings Limits

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Number of concurrent bindings	set max-users Command (see page 84)	255
Time that a DSA keeps a binding before closing it	set max-bind-time Command (see page 82)	No limit
Time that a binding can remain idle before being eligible to be closed.	set user-idle-time Command (see page 139)	3600
Time that a DSP binding between DSAs remains idle before the DSA closes it.	The DSP Idle Time setting in DXmanager	600

Multiwrite Replication Limits

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Time between attempts to reconnect to a multiwrite peer DSA if a connection is lost	set multi-write-retry-time Command (see page 87)	60
Maximum number of updates that can be held in a multi-write queue If this limit is exceeded, the DSA disables the queue, and the destination DSA will need to be manually resynchronized.	set multi-write-queue Command (see page 87)	20000

User Account Limits

A user account is an entry that contains a *userPassword* attribute.

You can only use the commands listed here if you have enabled password policies, with the set password-policy Command (see page 106).

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Time an account can be used before the password expires	set password-age Command (see page 92)	No limit
Maximum number of grace logins with an expired password	set password-grace-logins Command (see page 98)	No limit
Maximum time an account can remain unused before it becomes suspended	set password-last-use Command (see page 99)	No limit
Maximum number of failed logins with an incorrect password before an account becomes suspended	set password-retries Command (see page 107)	No limit
Maximum time an account will remain suspended before allowing logins	set password-max-suspension Command (see page 101)	No limit
Minimum time before a new password can be modified	set password-min-age Command (see page 103)	No limit
Maximum number of passwords to retain for reuse checking	set password-history Command (see page 98)	No limit
Minimum length of a new password	set password-min-length Command (see page 103)	6
Minimum number of certain characters allowed in a new password	set password-alpha Command (see page 95) set password-alpha-num Command (see page 95) set password-lowercase Command (see page 99) set password-non-alpha Command (see page 105) set password-non-alpha-num Command (see page 105) set password-numeric Command (see page 106) set password-uppercase Command (see page 109)	No limit

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
The maximum length of a new password	set password-max-length Command (see page 100)	No limit
The maximum number of times a single character can be repeated	set password-max-repetition Command (see page 100)	No limit
The maximum number of times a string can be repeated	set password-max-substring-repetition Command (see page 100)	No limit
The minimum length of the string used by password-max-substring-repetition	set password-min-length-repeated-substring Command (see page 104)	2

Logging Limits

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Minimum duration of the compare and search operations that you want to see in the time log	set time-log-search-threshold Command (see page 132)	No limit
Minimum duration of the add, modify, moddn, and remove operations that you want to see in the time log	set time-log-update-threshold Command (see page 133)	No limit

Other Limits

What Is Limited	How to Set the Limit	Default If Limit Is Not Changed
Size of a protocol data unit acceptable by DSAs	set max-pdu-size Command (see page 84)	0 (unlimited)

Limits That You Cannot Change

What is Limited	Description	Value
Number of entries in a DSA	CA Directory can cope with enormous numbers of entries.	More than one hundred million
Number of DSAs in a backbone	CA Directory can cope with enormous numbers of DSAs.	Hundreds
Number of replica DSAs	CA Directory can cope with enormous numbers of DSAs.	Hundreds
Number of attribute values per attribute in an entry	This is the maximum number of attribute values in a particular attribute in a single entry. This is not the overall number of values that exist in the directory for that attribute type. **Note:** Performance will degrade significantly (especially for updates) if some entries have large numbers of attribute values.	4,294,967,295
Size of RDNs	There is no restriction on the length of RDNs.	No limit
Depth of DIT stored in a single DSA	The maximum depth of a DIT stored in a single DSA is determined by the number of entries in the DSA.	19-56
Characters in the name of a DSA, computer, or user	See the Readme	

Data Limits

What is Limited	Description	Value
Number of searchable characters in an attribute value	There is no restriction on the number of searchable characters in DSAs.	
Number of characters in commands	UNIX shells and the Windows command line may limit the number of characters in a single command. This can a problem if you want to set the installation destinations for many CA Directory components. To avoid using a very long command, use the ETRDIRBASEPATH argument to set the location for the entire installation, instead of setting the location of each component separately.	
Maximum attribute value size	This is sometimes called the Maximum Binary Large Object (BLOB) size.	59 MB
Size of attribute values in an LDAP request	The largest LDAP request that a DSA can accept is limited.	30 MB
Size of integer attribute values	The largest integer syntax value is limited.	4 B
Number of attribute values per attribute in an entry	This is the maximum number of values in a particular attribute in a single entry. **Note:** Performance will degrade significantly (especially for updates) if some entries have large numbers of attribute values.	More than one hundred million

Namespace Limits

What is Limited	Description	Value
Size of RDNs	There is no restriction on the length of RDNs.	No limit
Depth of namespace tree in a single DSA	The maximum depth of a Namespace in a single DSA is determined by the number of entries in the DSA.	19-56

DSA Limits

What is Limited	Description	Value
Number of entries in a DSA	CA Directory can cope with enormous numbers of entries.	More than one hundred million
Number of DSAs in a backbone	CA Directory can cope with enormous numbers of DSAs.	Hundreds
Number of replicated DSAs in a backbone	CA Directory can cope with enormous numbers of replicated DSAs.	Hundreds

Other Limits

What is Limited	Description	Value
Characters in the name of a file, computer, or user	See the Readme	

Chapter 8: Supported Schemas and Syntaxes

This section contains the following topics:

Schema Files Provided with CA Directory

CA Directory provides a full directory schema definition starter kit, including:

- OSI (X.500, X.400, X.435 [EDI])
- Internet (LDAP, Inet, Cosine, Quipu, Thorn)
- Industry (JNDI, DADF, Mosaic, UNSPSC)
- Organizations (DMS, Umich, Entrust, RSA, Netscape)

Some of the schemas are referenced in *config/schema/dxmanager.dxg*. If you source this file in a DSA's initialization file, the schemas listed will be available to the DSA.

The following table lists the schema files that are supplied in the *config/schema* folder.

File Name	Group	Description
components.dxc	CA	Component fields within a certificate
cosine.dxc	Internet	RFC 1274 - The COSINE and Internet X.500 schema
dms.dxc	Organization	Defense Messaging Standard
dxserver.dxc	OSI	X.500 operational attributes and password management attributes
e3x.dxc	Internet	Experimental classes for French Gastronomique Review
edi.dxc	OSI	EDI Standard from ISOCOR
entrust.dxc	Industry	Entrust schema
inetop.dxc	Internet	Internet Organizational Person - RFC 2798

File Name	Group	Description
iplanet.dxc	Industry	iPlanet Directory schema
isocor.dxc	Industry	ISOCOR schema
iTechpoz.dxc	CA	Embedded Entitlements Manager schema
jndi.dxc	Industry	JAVA Naming and Directory Interface
ldapv3.dxc	Internet	LDAP v3 schema framework
lightship.dxc	Industry	Lucent Lightship
mhs.dxc	OSI	X.400 Message Handling System - RFC 1327/RFC 2156
mosaic.dxc	Industry	Netscape Mosaic Schema
nadf.dxc	Industry	North American Directory Forum
nisschema.dxc	Internet	NIS schema - RFC 2307
nsroaming.dxc	Industry	Netscape Navigator's Roaming Option
odmedia.dxc	CA	Support for multimedia
pp.dxc	OSI	X.400 standard
quipu.dxc	Internet	QUIPU schema
rfc2307bis.dxc	Internet	LDAP as a Network Information Service
rsa-keon.dxc	Industry	RSA Keon schema
rsa-pkcs9.dxc	Industry	RSA PKCS9 schema
sunone.dxc	Industry	Sun's Directory Server
thorn.dxc	Internet	Thorn schema
umich.dxc	Organization	University of Michigan schema
unspsc.dxc	Industry	United Nations Standards Products and Services

File Name	Group	Description
worldtalk.dxc	Industry	Worldtalk X.500 schema
x500.dxc	OSI	X.500 Standard schema

Supported Attribute Syntaxes

See the configuration files (.dxc) in the schema directory for the latest supported syntaxes.

All attribute syntaxes defined in X.520 are supported.

CA Directory also supports some attribute syntaxes defined in ACP 133. These are marked with * in the list below.

The supported attribute syntaxes are:

- audio
- binary
- bitString*
- boolean
- caseExactIA5String
- caseExactString
- caseIgnoreAlphaString*
- caseIgnoreIA5String
- caseIgnoreList
- caseIgnoreString
- certificate
- certificateList
- distinguishedName
- enumerated*
- facsimileNumber
- fax
- generalizedTime
- guide
- integer
- jpeg
- mhsDLSubmitPermission*
- mhsORAddress*
- mhsORName*
- nameAndOptionalUID
- numericString
- objectIdentifier
- octetString
- postalAddress
- preferredDelivery
- presentationAddress
- printableString
- printableStringList*
- rIParameters*
- telephoneNumber
- teletexTerminalId
- telexNumber
- uTCTime

More information:

set syntax-alias Command (see page 132)

Chapter 9: Supported Standards and Protocols

This section contains the following topics:

Conformance with Standards

This section contains information about CA Directory conformance with standards.

X500 Standards (see page 262)

CA Directory supports all the mandatory requirements of these standards.

LDAP Standards (see page 264)

CA Directory supports all of the significant LDAP Requests for Comments.

Management Standards (see page 268)

CA Directory supports SNMP and Telnet.

Security Standards (see page 270)

CA Directory supports SSL/TLS.

Web Services Standards (see page 271)

CA Directory supports DSML, and SOAP.

The detailed conformance of the CA Directory DSA with DAP, DSP, DISP, and LDAP protocols are documented in the Protocol Implementation Conformance Statement (PICS) and Internationalisation Profiles (ISPs) response document.

X.500 Standards

X.500 is a series of networking standards covering directory services.

This section contains the following topics:

Standards

CA Directory supports all the mandatory requirements of the following standards:

Recommendation	Title
X.500, ISO/IEC 9594-1 (1993)	Information technology - Open Systems Interconnection - The Directory: Overview of Concepts, Models, and Services
X.501, ISO/IEC 9594-2 (1993)	Information technology - Open Systems Interconnection - The Directory: Models
X.511, ISO/IEC 9594-3 (1993)	Information technology - Open Systems Interconnection - The Directory: Abstract Service Definition
X.518, ISO/IEC 9594-4 (1993)	Information technology - Open Systems Interconnection - The Directory: Procedures for Distributed Operation
X.519, ISO/IEC 9594-5 (1993)	Information technology - Open Systems Interconnection - The Directory: Protocol Specifications
X.520, ISO/IEC 9594-6 (1993)	Information technology - Open Systems Interconnection - The Directory: Selected Attribute Types
X.521, ISO/IEC 9594-7 (1993)	Information technology - Open Systems Interconnection - The Directory: Selected Object Classes
X.509, ISO/IEC 9594-8 (1993)	Information technology - Open Systems Interconnection - The Directory: Authentication Framework
X.525, ISO/IEC 9594-9 (1993)	Information technology - Open Systems Interconnection - The Directory: Replication
X.501 Corrigendum 1 (03/2000)	Information technology – Open Systems Interconnection – The Directory: Models Technical Corrigendum 1
X.501 Corrigendum 2 (02/2001)	Information technology – Open Systems Interconnection – The Directory: Models Technical Corrigendum 2

PICS

The Protocol Implementation Conformance Statement (PICS) documents consist of the main document plus four attachments detailing how CA Directory DSAs conform with DAP, DSP, DISP, and LDAP.

Proforma	Title	Comments
Recommendation X.586-6/ ISO/IEC 15126	CA Protocol Implementation Conformance Statements (PICS) and International Standardization Profiles (ISPs) for CA Directory.	**PICS Main** This is the PICS and Internationalization Profiles (ISPs) response document for CA Directory. All responses are regarding CA Directory DSAs. The DUA sections were left empty intentionally.
Recommendation X.583, ISO/IEC 13248-1 (1997 E)	CA Response to: Information Technology — Open Systems Interconnection — Directory Access Protocol: Protocol Implementation Conformance Statement (PICS) Proforma Recommendation X.583 - ISO/IEC 13248-1 for the eTrust Directory - Directory System Agent	**PICS Attachment 1** This is the Directory Access Protocol (DAP) PICS response attachment for the Directory System Agent (DSA) as per recommendation X.583 – ISO/IEC 13248-1 for CA Directory
Recommendation X.584, ISO/IEC 13248-2 (1997 E)	CA Response to: Information Technology — Open Systems Interconnection — Directory System Protocol: Protocol Implementation Conformance Statement (PICS) Proforma Recommendation X.584 - ISO/IEC 13248-2 for the eTrust Directory - Directory System Agent	**PICS Attachment 2** This is the Directory System Protocol (DSP) PICS response attachment for the Directory System Agent (DSA) as per recommendation X.584 – ISO/IEC 13248-2 for CA Directory.
Recommendation X.586, ISO/IEC 13248-4 (1997 E)	CA Response to: Information Technology — Open Systems Interconnection — Directory Information Shadowing Protocol: Protocol Implementation Conformance Statement (PICS) Proforma Recommendation X.586 - ISO/IEC 13248-4 for the eTrust Directory - Directory System Agent	**PICS Attachment 3** This is the Directory Information Shadowing Protocol (DISP) PICS response attachment for the Directory System Agent (DSA) as per recommendation X.586 – ISO/IEC 13248-4 for CA Directory.

Proforma	Title	Comments
CEN/ISSS Directory Workshop Draft 5	CA Response to: CEN/ISSS Directory Workshop — Lightweight Directory Access Protocol V3: Level of Support of an LDAP Server Draft 5 – 25.5.1998 for the eTrust Directory - Directory System Agent.	**PICS Attachment 4** This is the Lightweight Directory Access Protocol (LDAP) PICS response attachment for the Directory System Agent (DSA) as per CEN/ISSS Directory Workshop Draft 5 document for CA Directory.

LDAP Standards

CA Directory supports all of the significant LDAP Request for Comments (RFCs). Support for other LDAP standards will be included as they gain industry acceptance.

This section contains the following topics:

LDAP RFCs

CA Directory supports all the significant LDAP Requests for Comment (RFCs). These are listed in the following table:

RFC	Title	Obsoletes RFCs	Status
1777	Lightweight Directory Access Protocol (March 1995)	1487	Draft standard
1778	The String Representation of Standard Attribute Syntaxes (March 1995)	1488	Updated by RFC2559 Draft standard
2696	LDAP Control Extension for Simple Paged Results Manipulation		Informational
2891	LDAP Control Extension for Server Side Sorting		Proposed standard Server-side sorting is only supported for attributes configured for caching.

RFC	Title	Obsoletes RFCs	Status
4510	Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map (June 2006)	2251, 2252, 2253, 2254, 2255, 2256, 2829, 2830, 3377, 3771	Proposed standard

Internet Draft Standards

To read these standards, check on the IETF home page.

Draft	Title	Comments
Behera	Password Policy for LDAP Directories (October 2004)	Support LDAP Controls only Status: Expired April 2005
Dally	ACP 133 Common Content and LDAP (22 September 2000)	CA Directory supports only a subset of the syntaxes Status: Expired March 2001
Persistent Search	LDAP Persistent Search (November 2000) - includes Entry Change Notification	Status: Expired May 2001
Weltman	LDAP Proxied Authorization Control June 2005)	Status: Expired December 2005

PICS

The LDAP PICS is listed with the X.500 PICS (see page 263).

Conformance Tests

The LDAP protocol is tested with the following conformance tests:

Test Suite	Description
BLITS	CA Directory has been tested against the Basic LDAP Version 3 Interoperability Test Suite. For more information, see http://www.opengroup.org/dif/blitspub/blits3.0/.
PROTOS	CA Directory has been tested against the PROTOS protocol security test suite for LDAP, which was made prominent by CERT. See the CERT Advisory page.
VSLDAP	CA Directory has been tested against the VSLDAP test suite. VSLDAP is The Open Group's test suite for servers of the Lightweight Directory Access Protocol (LDAP) Version 3. See http://www.opengroup.org/dif/ldapc/index.htm for more information.

Identrus Certifications

CA Directory is fully certified by Identrus for authentication of digital identities.

See http://www.identrust.com/company/press_releases/archives/release_030709.html

Open Group Certifications

The purpose of the Open Group LDAP Certified standard is to certify that CA Directory servers interoperate with LDAP V3 clients. The standard has two main conformance requirements:

BASE profile

This is a compulsory requirement

STANDARD profile

This is an optional requirement, but the conformance statement must state whether the product implements the features of the STANDARD profile.

eTrust Directory r8.1 is 2.2 LDAP certified V2.2 and implements both the BASE and STANDARD features.

The following table lists the platforms on which eTrust Directory r8.1 is certified:

Platform	Link to Brand Certificate
Solaris 10 on Sun Blade 1500	http://www.opengroup.org/openbrand/certificates/0766c.pdf
Solaris 10 (x86) on Sun V20Z (Opteron CPU)	http://www.opengroup.org/openbrand/certificates/0766c.pdf
Windows XP on Dell GX270	http://www.opengroup.org/openbrand/certificates/0767c.pdf
AIX 5L v5.3 on AIX Power PC (P5)	http://www.opengroup.org/openbrand/certificates/0769c.pdf
Linux Red Hat 4.1 on Dell Poweredge 6400	http://www.opengroup.org/openbrand/certificates/0768c.pdf

All the Brand Certificates have the same dates:

- Registered: 12 January 2006
- Valid Until: 12 January 2008

Industry Extensions

CA Directory supports the following industry extensions made popular by other LDAP products:

Test Suite	Source
Class of Service	http://docs.sun.com/source/816-5583-10/06_cos.htm
Dynamic Groups	http://www-1.ibm.com/servers/eserver/iseries/ldap/dynamicgroup.htm

Management Standards

CA Directory supports the following industry standards for management.

This section includes the following topics:

ISO Standards

The CA Directory supports the following industry standards for management:

Standard	Title
Recommendation X.711	Data communication networks: Open Systems interconnection (OSI); Management Common Management Information Protocol Specification for CCITT applications (1991)
Recommendation X.720 ISO/IEC IS 10165-1	Information technology - Open Systems Interconnection - Structure of Management Information - Part 1: Management Information Model
ISO/IEC CD 9594-10	Information technology - Open Systems Interconnection - The Directory: Use of Systems Management for Administration of the Directory

ISO Corrigendum

Recommendation	Title
X.711 Corrigendum 1 (03/1999)	Information technology – Open Systems Interconnection – Common management information protocol: Specification Technical Corrigendum 1
X.711 Corrigendum 2 (02/2000)	Information technology – Open Systems Interconnection – Common management information protocol: Specification Technical Corrigendum 2: Revision to include ASN.1: 1997 ITU-T

Management RFCs

RFC	Title	Comments
854	Telnet Protocol Specification (May 1983)	Obsoletes: NIC 18639 and STD0008 Status: Standard
1155	Structure and Identification of Management Information for TCP/IP-based Internets (May 1990)	Obsoletes RFC1065 Also STD0016 Status: Standard
1157	Simple network management protocol(SNMP). (May 1990)	Obsoletes RFC1098 Also STD0015 Status: Standard
1212	Concise MIB Definitions (March 1991)	Obsoletes STD0016 Status: Standard
1213	Management Information Base for Network Management of TCP/IP-based internets: MIB-II (March 1991)	Obsoletes RFC1158) Updated by RFC2011, RFC2012, RFC2013, and STD0017 Status: Standard
1567	X.500 Directory Monitoring MIB (January 1994)	Obsoleted by RFC2605 Status: Proposed standard

Security Standards

Standard	Title	Comment
SSL V3.0	The SSL Protocol Version 3.0 (Nov 1996)	Status: IETF Internet Draft
PKCS #11	PKCS #11 v2.11: Cryptographic Token Interface Standard (November 2001)	This is an "RSA Security Inc. Public-Key Cryptography Standards (PKCS)"
PKCS #12	PKCS 12 v1.0: Personal Information Exchange Syntax Standard (June 1999)	This is an "RSA Security Inc. Public-Key Cryptography Standards (PKCS)"
RFC 2246	The TLS Protocol Version 1.0	Status: Proposed standard
PKCS #11A	PKCS #11 v2.11 Amendment 1 (August 2002)	This is an "RSA Security Inc. Public-Key Cryptography Standards (PKCS)"
PKCS #12TC	PKCS #12 v1.0 Technical Corrigendum (February 2000)	This is an "RSA Security Inc. Public-Key Cryptography Standards (PKCS)"
FIPS 140-2	SECURITY REQUIREMENTS FOR CRYPTOGRAPHIC MODULES	CA Directory r12 and later utilizes an embedded cryptographic module that has been validated as meeting the Federal Information Processing Standards (FIPS) 140-2 Security Requirements for Cryptographic Modules. This module (RSA BSAFE® Crypto-C ME) provides all of the cryptographic services in the CA Product. The validation certificate number for this module is #608.
FIPS 186-2	DIGITAL SIGNATURE STANDARD (DSS) 2000 January 27	CA Directory r12 and later utilizes an embedded cryptographic module (RSA BSAFE® Crypto-C ME) that has been validated as meeting FIPS 140-2 Security Requirements for Cryptographic Modules. The validation for FIPS140-2 includes certification of DSA (Cert. #143); RNG (Cert. #130); ECDSA (Cert. #11). These algorithms are required by FIPS 186 and hence supports the implementation of a DSS solution.

Web Services Standards

Standard	Title	Comment
DSML 2.0	Directory Services Markup Language v2.0 (December 2001)	This document is an approved OASIS Standard April 30, 2002. CA Directory includes support for the DSML 2.0 connection protocol. DSML is a configurable option and it runs on a separate port from those of LDAP and X.500. See the DSML TC page.
SOAP 1.1	Simple Object Access Protocol (SOAP) 1.1 (W3C Note 08 May 2000)	CA Directory DSML implements the widely used Simple Object Access Protocol (SOAP) standard for communication to minimize integration effort. See the SOAP Note.

Errata	Title
DSML 2.0	Directory Services Markup Language v2.0 Errata

Hashing Formats

For explanations of the cryptographic terms used in the following sections, refer to TLS Request For Comments (RFC) 2246:

- Hashing formats for DXserver passwords
- Hashing formats for the DSA console password
- Encryption formats for SSL (see page 273)

Hashing Formats for DXserver Passwords

CA Directory can store user passwords in the following formats:

SHA

(Default) Hashes the password using the SHA-1 algorithm.

SSHA

Hashes the password using the Salted SHA-1 algorithm. This algorithm produces a different hash even for the same clear text password, which is more secure.

SHA512

Hashes the password using the SHA-512 algorithm.

MD5

Hashes the password using the Message Digest algorithm.

CRYPT

Hashes the password using the UNIX *crypt* method.

CADIR

Hashes the password using a reversible obfuscation algorithm. Use this option to hash a password before including it in a knowledge file in the *dsa-password* or *ldap-dsa-password* configuration items. This protects the password from users with access to the computer running the DSA.

For more information, see Password Storage.

Hashing Formats for the DSA Console Password

The DSA console password can be hashed using the following formats:

SHA

(Default) Hashes the password using the SHA-1 algorithm.

SSHA

Hashes the password using the Salted SHA-1 algorithm. This algorithm produces a different hash even for the same clear text password, which is more secure.

SHA512

Hashes the password using the SHA-512 algorithm.

MD5

Hashes the password using the Message Digest algorithm.

CRYPT

Hashes the password using the UNIX *crypt* method.

CADIR

Hashes the password using a reversible obfuscation algorithm. Use this option to hash a password before including it in a knowledge file in the *dsa-password* or *ldap-dsa-password* configuration items. This protects the password from users with access to the computer running the DSA.

Encryption Formats for SSL

To protect communications links, CA Directory can use SSL encryption. The supported encryption techniques are listed below.

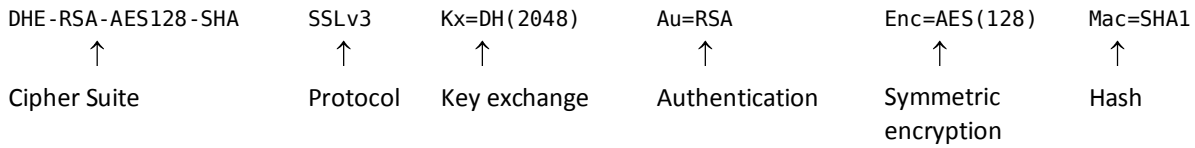
Supported Cipher Suites

To list supported cipher suites, use the following console command:

```
get ciphers
```

This command lists the cipher suites supported by CA Directory. Each row in the list describes one supported cipher.

For example, the following row in the output describes the DHE-RSA-AES128-SHA cipher suite:



Supported Key Exchange Algorithms

The following table lists the key exchange algorithms supported by CA Directory:

Exchange	Description of Algorithm	Key Size Limit
DHE_DSS	Ephemeral DH with DSS signatures	DH = 2048 bits
DHE_DSS_EXPORT	Ephemeral DH with DSS signatures	DH = 512 bits
DHE_RSA	Ephemeral DH with RSA signatures	DH = 2048 bits
DHE_RSA_EXPORT	Ephemeral DH with RSA signatures	DH = 512 bits
DH_anon	Anonymous DH, no signatures	DH = 2048 bits
DH_anon_EXPORT	Anonymous DH, no signatures	DH = 512 bits
DH_DSS	DH with DSS-based certificates	DH = 2048 bits
DH_DSS_EXPORT	DH with DSS-based certificates	DH = 512 bits
DH_RSA	DH with RSA-based certificates	DH = 2048 bits
DH_RSA_EXPORT	DH with RSA-based certificates	DH = 512 bits
RSA	RSA key exchange	RSA = 2048 bits
RSA_EXPORT	RSA key exchange	RSA = 512 bits

US Government Standards

The following sections describe the extent to which CA Directory conforms to standards and programs required by the US government.

Accessibility (Section 508)

Section 508 requires that US Federal agencies' electronic and information technology is accessible to people with disabilities. This includes two main areas:

Keyboard Access

To be able to operate all CA Directory functionality without using a mouse

Assistive Devices

To be able to operate all CA Directory functionality with assistive technology such as a screen reader

CA Directory conforms to Section 508 requirements as follows:

Component	Level of Conformance	Approval Date
DXserver All command-line driven DXtools	Supports 1194.21 (Software Applications and Operating Systems) Supports 1194.31 (Functional Performance Criteria) Does not support 1194.41 (Information, documentation, and support)	19 Oct 2005
DXmanager	Does not support Section 508	13 Oct 2005
JXweb	Supports with exception 1194.21 (Software Applications and Operating Systems) Supports with exception 1194.31 (Functional Performance Criteria) Does not support 1194.22 (Web-based internet information and applications) Does not support 1194.41 (Information, documentation, and support)	13 Oct 2005

Common Criteria Certification

The Common Criteria is a standard for evaluating information technology products and systems, such as operating systems, computer networks, distributed systems, and applications.

Its purpose is to allow users to specify their security requirements, to allow developers to specify the security attributes of their products, and to allow evaluators to determine if products actually meet their claims.

Version 2.1 of the Common Criteria is equivalent to the ISO International Standard 15408 [I15408].

CA Directory r8.1 has been certified by Cygnacom Solutions. We evaluated at Assurance Level EAL3.

FIPS Compliance

FIPS 140-2 (Security Requirements for Cryptographic Modules) is a security accreditation program for cryptographic modules in software that is used by government departments and industries that deal with "sensitive, but not classified" information. Validation is coordinated by the National Institute of Standards and Technology (NIST).

CA Directory utilizes an embedded cryptographic module that has been validated as meeting the Federal Information Processing Standards (FIPS) 140-2 Security Requirements for Cryptographic Modules.

See the NIST website at http://csrc.nist.gov/cryptval/140-1/140val-all.htm#608 for details of the RSA BSAFE® Crypto-C ME cryptographic services.

Link Protocols

CA Directory uses industry standard protocols for client-to-server and server-to-server communications. Any of the following protocols may be operated over an encrypted link.

For more information about securing communications with SSL, see Set Up Encryption.

This section contains the following topics:

Server–Server Links

The following table lists the link protocols used by CA Directory when linking to between directory servers:

Protocol	Full Name	Use of Protocol	Standards
DSP	Directory System Protocol	Server-to-server chaining	X.519
DISP	Directory Information Shadowing Protocol	Server-to-server replication	X.519
LDAP	Lightweight Directory Access Protocol	Server-to-server replication	RFC 4510

Client–Server Links

The following table lists the link protocols used by CA Directory when linking to clients:

Protocol	Full Name	Use of Protocol	Standards
LDAP	Lightweight Directory Access Protocol	Client access	RFC 4510
DAP	Directory Access Protocol	Client access	X.519
Telnet	Telnet	Management console	RFC 854

Web Services Protocols

CA Directory uses the OASIS standard DSML through a DSML-Tomcat intermediary.

IPv6 Support

IPv6 is the next generation of the Internet Protocol, designed not only to extend the address space (to cope with growth in Internet usage) but also to take advantage of concepts like multicast and quality of service.

All components of CA Directory fully support IPv6.

Protocol Diagrams

This section shows the components of CA Directory and the protocols that they use.

Diagram of Directory Protocols

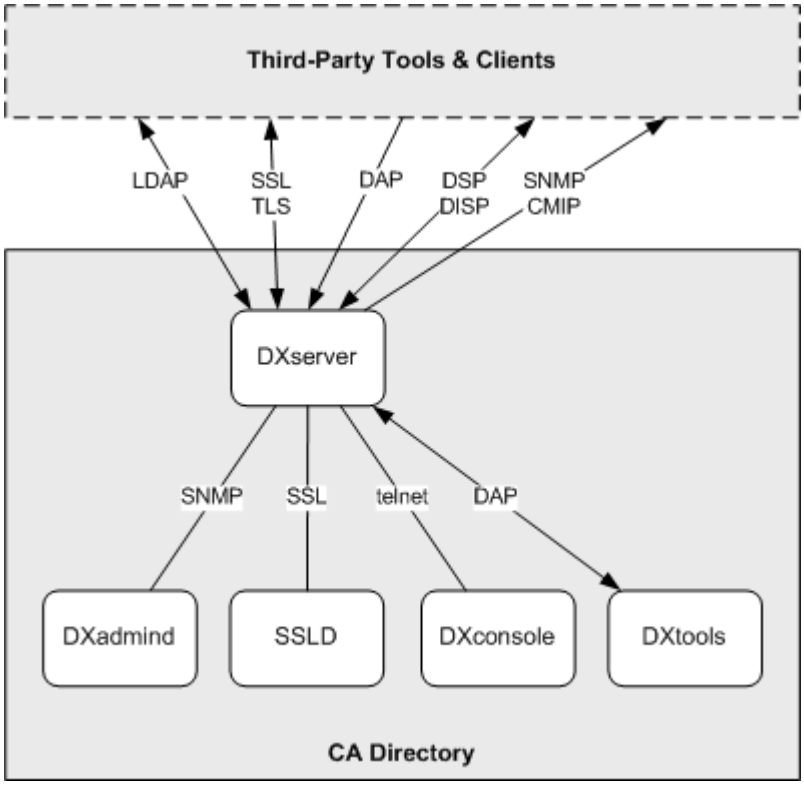
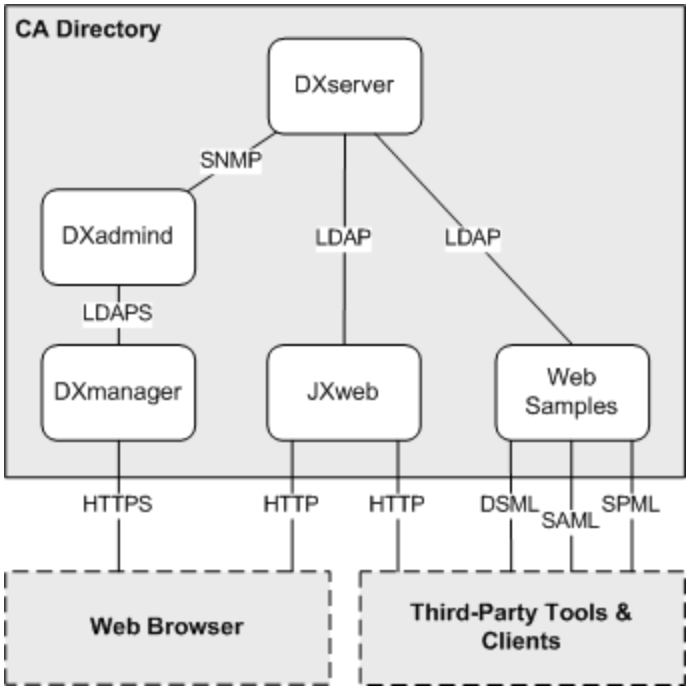


Diagram of Web Services Protocols



Chapter 10: Port Numbers Used by CA Directory

This section lists the port numbers used in a default CA Directory installation. If one of these port numbers is also used by another application on the same computer, you may need to change the port number used by CA Directory.

This section contains the following topics:

Port Number Range

When you are running CA Directory on Windows, you usually use a port number between 1700 and 64K (65536).

When you are running CA Directory on a UNIX platform, your system administrator will allocate available port numbers. On most systems, the port numbers also go up to 64K (65536).

NSAP, PSAP, and TSAP Port Numbers

You can use DXmanager to configure NSAP, PSAP, and TSAP ports. However, we recommend that you do not use these options.

These options are available to ensure interoperability with third-party X.500 directories. They are not required for LDAP.

This is because the X.500 standard was built on top of the OSI model, whereas LDAP was layered directly over TCP/IP. TCP/IP effectively maps to the Network and Transport layers, and the concept of ports negates the need for NSAPs and PSAPs.

Ports Used by CA Directory Components

The following table lists the port numbers used by the CA Directory components in a default installation.

Each DSA you create will take up at least one additional port number.

You can use the get stack command to see the port numbers used on a DSA.

Component	Protocol	Port Number	Description of Port	Accepts SSL Requests	Configuration
DXwebserver	TCP	8080	DXwebserver listens on this port for non-SSL connections.	No	Change DXwebserver's Port Numbers
	TCP	8443	DXwebserver listens on this port for SSL connections. This is also used for the Coyote/JK2 1.3 connector port.	Yes	
	TCP	8005	DXwebserver listens on this port for the shutdown command.	Yes	
DXadmind	TCP	2123	This port is used for LDAP communication between DXadmind and DXmanager.	No	This port number is not configurable.
	TCP	2125	DXadmind uses this secure LDAPS port to monitor remote DSAs. A user name and password are required, and connections are only accepted from trusted computers.	Yes	This port number is not configurable.
DXtrap Sample Tool	TCP	162	DXtrap receives SNMP traps on this port. For more information, see the Readme in DXHOME/samples/trap.		

Chapter 11: The Directory User Agent (DUA)

This section contains the following topics:

The DUA

The *directory user agent* (DUA) is an executable that accesses DSAs as a client. The DUA is supplied in the samples folder and can run on any host that has directory services installed. It communicates user requests to a DSA, which can be be on any host in the network, and then passes the DSA responses back to the user.

Usually you use an LDAP client such as JXweb to access the directory. However the DUA is useful when you want a command line interface, for example to run batch operations such as tests on the directory.

The DUA commands fully implement every aspect of every X.500 service and form the basis for testing CA Directory. You can consult the test scripts released with CA Directory for various examples of any given service.

DUA commands issued at the console are asynchronous. That is, you can enter a command without waiting for the previous one to be complete. If you use DUA commands in a script then use the wait command to force the script to wait till the command is complete.

The DUA prompt is *dua>*.

See the X.500 standards documentation for full details of Directory User Agent services.

Example Script for a DUA

Usually you put the DUA commands in a script file so you can run them in batch mode.

The following set of command initializes the DUA and connects it to the DSA specified by the NSAP parameter in the bind request.

Example: Script for a DUA

```
# DUA Script Sample
set summary-log = "DUAcli.log";   # keep a summary log
echo-on                           # output commands to log
# Include schema definitions.
source "../schema/schema.dxg";     # schema rules
# Log on
bind-req
        user = <countryName            "AU">
                <organizationName       "Democorp">
                <organizationalUnitName "Services">
                <organizationalUnitName "Networks">
                <commonName             "Brendan RANDALL"
        password   = "secret"
        remote-addr = {
                nsap = ip "192.9.200.1" port 19389
                } ;
wait;
if-reply bind-refuse then goto finish;
# List root entries, to show we're online
list-req
        entry = < > ;
wait;
if-reply list-refuse then goto finish;
#
#...Put other DUA commands
#
#close the binding
unbind req;
wait;
#......
% finish
flush summary-log;
quit;
```

DUA Commands

This section contains the following topics:

abandon-req Command—Stop a Request

This command stops a read type request. The stopped command returns any partial results that it had produced at the time it was stopped.

You can abandon the following requests:

- compare-req
- list-req
- read-req
- search-req

You cannot abandon an update request.

The format of this command is as follows:

```
abandon-req [invoke-id-to-abandon = invoke-id];
```

invoke-id

Specifies the ID number of the service to be abandoned.

An *abandon confirmed* message is returned if the request is stopped. If the request completes, or it cannot be abandoned, then an *abandon refuse* message is returned.

Example: Abandon Search 1234

```
abandon-req invoke-id-to-abandon = 1234;
```

add-entry-req Command—Add an Entry

The add-entry-req command adds an entry. The new entry must obey the name-binding rules, and the entry attributes must obey the entry's object-class rules.

The format of the command is as follows:

```
add-entry-req
        entry = DN
        contents = {
                (objectClass objectClass-name)
                (attribute attribute-value [,attribute-value...] )
                [...]
                }
        [common-args]
        ;
```

DN

Defines the entry to be added, expressed in x.500 format.

objectClass *objectClass-name*

Defines the object class of the new entry.

***attribute attribute-value* [*,attribute value...*]**

Defines an attribute and its value (or values). Multiple values are separated by commas. A new line within an attribute value is denoted by a period (.).

common-args

Defines the common arguments. For more information, see common-args (see page 289).

Example: Add a Single Entry

```
add-entry-req
        entry = <countryName "AU"> <organizationName "Democorp">
<organizationalUnitName "Sales">
        contents = {
                ( objectClass organizationalUnit )
                ( postalAddress "100 222–268 Maroondah
Highway"."Mooroolbark"."Victoria" )
                ( postalCode "3138" )
                ( telephoneNumber "(03) 9727-8900" , "(03) 9727-8901" )
                ( facsimileTelephoneNumber "(03) 9727-3491" )
        };
```

Example: Add an Entry with a Multivalue Naming Attribute

In this example, a multivalued attribute *commonName* has the value "John Smith", which is the distinguished name and names the entry. The *commonName* attribute is in the contents of the add request so that you can add the value J SMITH.

```
add-entry-req
        entry = <countryName "AU">  <organizationName "Democorp"> <commonName "John
Smith">
        contents = {
                ( objectClass organizationalPerson )
                ( surname "Smith" )
                ( commonName "J SMITH" )
        };
```

See the test scripts for more examples.

Add an Alias

To add an alias with the add-entry-req command, you supply the object class *alias* and the attribute *aliasedObjectName*.

Aliases have no name binding rules, so you can add an alias entry anywhere in a DIT.

If when you add an alias, the DSA has alias integrity enabled, the DSA must be able to navigate to the entry that the alias points to.

Example: Add an Alias Entry

```
add-entry-req
        entry = <countryname "AU">
                <organizationName "Democorp">
                <commonName "Brendan Randall">
        contents = {
                (objectClass alias)
                (aliasedObjectName
                        <countryname "AU">
                        <organizationName "Democorp">
                        <organizationalUnitName "Services">
                        <organizationalUnitName "Networks">
                        <commonName "Brendan Randall"> )
                }
        ;
```

bind-req Command—Create a Binding

You create a binding between the DUA and a DSA by using the bind-req command.

When you enter the bind-req command using the DSA console, the system treats the command as if it had come from a remote DUA.

A binding remains active until one of the following happens:

- It is released by the user using the unbind service.
- It is aborted, probably because the connection failed or the DSA was shut down.
- It is terminated by the DSA (probably because a time limit expired).
- It is terminated by the system administrator (using the *abort users* command).

The bind-req command has the following format:

```
bind-req {
        [ssl-auth [pem-file]]
        | [[ssl-encryption ] user=user password=password]
        }
        [ remote-addr= {
                rfc1278-p-addr
                | [psap=psap-addr] [ssap=ssap-addr][tsap=sap-addr][nsap= tcp
tcp-addr tcp-port port]
                }
        ]
```

ssl-auth

Specifies that the bind request should use SSL authorization.

pem-file

Defines the SSL personality file. Maximum length is 20 characters. The DUA adds the filename extension .pem.

ssl-encryption

Specifies that the username and password are to be encrypted.

Example: Binding Request:

```
bind-req
user = <countryName "AU">
        <organizationName "Democorp">
        <organizationalUnitName "Services">
        <organizationalUnitName "Networks">
        <commonName "Brendan RANDALL">
password = "secret"
```

Close a Binding

Close the binding between the DUA and the DSA by using one of the following methods:

- For outgoing bindings, you can use the unbind_req command, as follows:

  ```
  unbind-req;
  ```

  The DUA discards any unfinished operations and closes the binding.

- You can use the abort command, as follows:

  ```
  abort-req;
  ```

  The DUA drops all communication immediately.

- Close the TCP link to the DSA.

common-args—Common Arguments Applicable to All DUA Requests

The common-args parameter is common to all DUA requests. All common arguments are optional.

The format of the parameter is as follows:

```
common-args = {
        [chaining-prohibited]
        [dont-deref-aliases]
        [local-scope]
        [size-limit = numberOfEntries]
        [time-limit = numberOfSeconds]
        }
```

chaining-prohibited

Has the same effect as local-scope.

dont-deref-aliases

Specifies that the DUA should treat an alias as the target for the command.

The default for the commands that read entries is to treat an alias as a pointer, and return information about the entry that the alias points to. This applies to the following commands:

- compare-req
- list-req
- read-req
- search-req

By contrast, the commands that write to the directory always act on the DN specified in the command, and do not treat an alias as a pointer. Hence this option is implied. The update commands are the following:

- add-entry-req
- mod-dn-req
- mod-entry-req
- rem-entry-req

local-scope

Prevents the DSA that receives a request from passing the request to another DSA. Do not use this in a configuration with router DSAs, because a router DSA always needs to pass requests to another DSA. However, this parameter can be useful for testing purposes.

size-limit= *numberofEntries*

Defines the maximum size allowed for the returned result. If the request is not complete within this limit, it returns the partial results. If the max-op-size setting in the DSA configuration is smaller than size-limit, then max-op-size takes precedence.

This argument is only relevant for the inquire services: compare-req, read-req, search-req and list-req. It is ignored for the other, update, commands.

time-limit= *NumberOfSeconds*

Defines the maximum time the request is allowed. If the request is not complete within this time, it returns the partial results. If the max-op-time setting in the DSA configuration is smaller than time-limit, then max-op-time takes precedence.

compare-req Command—Compare Entries

The compare-req command compares the contents of an entry with some static data.

This command has the following format:

```
compare-req entry=DN

        assertion = attribute-name "attribute-value"

        [common-args]

        ;
```

entry=*DN*

Defines the distinguished name of the entry. *DN* is expressed in X.500 format.

assertion = *attribute-name* "*attribute-value*"

Specifies the name and value of the attribute to be compared.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

If the DSA cannot find the entry or attribute, it returns the following result:

```
compare-refuse
```

If DSA finds the entry and attribute, it returns one of the following results:

```
compare-confirm was-matched
compare-confirm not-matched
```

Example: Compare a Telephone Number

Use the following command to compare a telephone number:

```
compare-req
        entry = <c "AU"><organizationName "Democorp">
                <organizationalUnitName "Corporate">
        assertion = telephoneNumber "03 9727 9942";
```

See the test scripts for more examples.

list-req Command—List Entries

The list-req command displays the RDNs of the objects immediately under the list object.

When the DSA has knowledge of other DSAs, the list makes these references visible.

This command has the following format:

```
list-req

        entry = DN

        [common-args]

        ;
```

entry=*DN*

Defines the distinguished name of the entry. *DN* is expressed in X.500 format.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

Example: List Entries below the Organizational Unit Corporate under the Organization Democorp:

```
list-req entry = <countryName "AU">
        <organizationName "Democorp">
        <organizationalUnitName "Corporate">;
```

Example: List Entries below root:

```
list-req entry = <>;
```

mod-dn-req Command—Rename an Entry

The mod-dn-req command renames an entry. Renaming a non-leaf entry changes the distinguished name of all entries under it.

You can use mod-dn-req to change the case of a value, for example, from Democorp to DEMOCORP.

You can also optionally delete the old RDN.

The command has the following syntax:

```
mod-dn-req
        entry = DN
        new-rdn = DN
        [ delete-old | dont-delete-old ]
        [new-superior = DN ]
        [common-args]
        ;
```

delete-old

Specifies that the command should delete the old DN of the entry. If you do not specify delete-old, the RDN remains in the entry as one or more non-distinguished attribute values.

new-superior = *DN*

Specifies that the command should move the entry (and any subordinates) to a different place in the DIT.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

Example: Change the Name of an Organizational Unit

To change the name of the organizational unit *R&D* (under organization Democorp) to *Research & Development:*

```
mod-rdn-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "R&D" >
        new-rdn = <organizationalUnitName "Research & Development">
        delete-old;
```

Example: Move the Entry and Any Subentries

The following command moves the *R&D* entry and all of its subordinates from AU,Democorp to AU, Democorp, Corporate.

```
mod-rdn-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "R&D" >
        new-rdn = <organizationalUnitName "Research & Development">
        delete-old
        new-superior = <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName Corporate>;
```

mod-entry-req Command—Modify an Entry

The mod-entry-req command adds and removes attributes and values. If you use this command on an alias, it changes the alias, not the entry that the alias points to.

Note: You cannot use mod-entry-req to create an alias—use add-entry-req instead.

The command has the following format:

```
mod-entry-req
        entry = DN
        Modifications
        [common-args]
```

Modifications

Defines the changes to be made to the entry. The format of the modifications options is as follows:

```
add-attr {attribute attribute-value [,attribute-value...]}
| add-values {attribute attribute-value [,attribute-value...]}
| rem-values {attribute attribute-value [,attribute-value...]}
| rem-attr attribute
[, ...]
```

The braces ({}) in the *add-attr*, *add-values*, and *rem-values* options are part of the command.

If two or more modifications are specified, they are separated by commas.

add-attr

Specifies that an attribute is to be added. If the attribute already exists, *add-attr* is treated as if it were *add-values*.

add-values

Specifies that one or more values are to be added to an existing attribute. You cannot add a second value to a single-valued attribute.

rem-values

Specifies that one or more values are to be removed from the attribute. Removing the last value removes the attribute.

rem-attr

Specifies that an attribute is to be removed from the entry. You cannot remove mandatory attributes.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

Example: Modify Service to Add Attributes

Add a fax number to organizational unit Corporate under organization Democorp:

```
mod-entry-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "Corporate" >
        add-attr {facsimileTelephoneNumber "03-9727-9722" }
        ;
```

Example: Modify Service to Add and Remove Attributes

Add another phone number value for John Smith, and remove one of his common names—the non-distinguished value, J SMITH:

```
mod-entry-req
        entry = <countryName "AU">
                        <organizationName "Democorp">
                        <commonName "John Smith">
        rem-values { commonName "J SMITH" },
        add-values { telephoneNumber "03 9727 9111" };
```

See the test scripts for more examples.

Make an Alias Point To a Different Entry

You can use the mod-entry-req command to make an alias point to a different entry.

Note: You cannot change a non-alias entry into an alias.

If the DSA has alias integrity enabled when you add an alias, the DSA must be able to navigate to the object that the alias points to.

To modify an alias, use the mod-entry-req command to remove the value of the attribute *aliasedObjectName*, and add a new value.

Example: Change an Alias of a Person to be an Alias of a Role

```
mod-entry-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <commonName "Brendan Randall">
        rem-values {
                (aliasedObjectName
                <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "Services">
                <organizationalUnitName "Networks">
                <commonName "Brendan Randall"> )},
        add-values {
                (aliasedObjectName
                <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "Services">
                <organizationalUnitName "Networks">
                <commonName "System Manager">    }

        ;
```

Instead of removing and adding values, you can remove and add the attributes. This is probably the more common way to making such a change.

rem-entry-req Command—Delete an Entry

Use the rem-entry-req command to remove a leaf entry from the directory.

If the leaf entry is an alias, the alias is removed.

If the DSA has alias integrity enabled, when you delete an entry all aliases of the entry are also deleted.

You can delete a subtree by deleting leaves recursively, or by using another tool, such as the DXmodify tool.

The command has the following format:

```
rem-entry-req
        entry = DN
        [common-args]
        ;
```

entry=*DN*

Defines the distinguished name of the entry. *DN* is expressed in X.500 format.

common-args = {*common-args*}

See the description of common -args in read-req Command.

Example: Remove Service for Organizational Unit

To remove the organizational unit *Sales* from the organization *Democorp*:

```
rem-entry-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <organizationalUnitName "Sales">;
```

Example: Delete an Alias That Points to an Entry

```
rem-entry-req
        entry = <countryName "AU">
                <organizationName "Democorp">
                <commonName "Brendan Randall">;
```

read-req Command—Read an Entry

Use the read-req command to get information about an entry.

The command has the following format:

```
read-req entry = DN
        [no-info-return | attrs-only | attr-and-val]
        [all-attrs | attrs = attribute[,attribute...] ]
        [ all-extra-attrs | extra-attrs = attribute[,attribute...] ]
        [common-args]
        ;
```

entry=*DN*

Defines the distinguished name of the entry. *DN* is expressed in X.500 format.

attrs=*attribute* [,*attribute*]

Defines the non-operational attributes to display. The service returns an error if it does not find these attributes in the entry.

By default, all non-operational attributes are displayed.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

extra-attrs=*attribute* [,*attribute*]

Defines the operational attributes to display. By default no operational attributes are displayed.

Example: Get Information About the Democorp Entry

The following command returns all the attribute values for the Democorp entry:

```
read-req entry = <countryName "AU"><organizationName "Democorp"> ;
```

Example: Read Selected Attributes and Values

The following command returns the selected attribute values for the Corporate entry:

```
read-req
        entry = <c "AU"><o "Democorp"><ou "Corporate">
        attrs = facsimileTelephoneNumber, telephoneNumber;
```

Example: Read an Alias Object, not the Entry that the Alias Points To

```
read-req
        entry = <countryName "AU">
                <organizationName "Democorp">

                <commonName "Brendan Randall">
                common-args = {dont-deref-aliases } ;
```

search-req Command—Find Entries

The search-req command finds entries within a namespace partition.

The search service can return many objects. If the search exceeds a size limit or a time limit, or if the DUA receives an abandon request, the search returns partial results, that is, the entries that it collected before being stopped.

The search is successful if it can find its base object.

The search detects if an alias directly or indirectly points to itself, and returns an error.

The command has the following format:

```
search-req
        base-object = DN
        [one-level-only | base-object-only | whole-subtree]
        [filter = { search-req-filter }]
        [no-info-return]
        [dont-search-aliases | search-aliases]
        [attr-only | attr-and-val]
        [all-attrs | attrs = attribute [,attribute...]]
        [common-args]
        ;
```

base-object = *DN*

Defines the distinguished name of the entry at which the search starts. *DN* is expressed in X.500 format.

one-level-only | base-object-only | whole-subtree

(Optional) Specifies the scope of the search.

The default is base-object-only.

{*search-req-filter* }

Specifies a filter for the search. The braces are part of the command. For more information about its syntax, see search-req-filter (see page 301).

no-info-return

(Optional) Returns only the names of the entries that satisfy the filter.

dont-search-aliases

(Optional) Specifies that search-req should not follow the aliasedObjectname of an alias. If this is not specified, search-req treats aliases as pointers to entries.

attrs = *attribute* [,*attribute*...]

(Optional) Returns only the specified attributes.

common-args

Defines the common arguments. For more information, see common-args (see page 289).

Example: Search Service for One Entry

The following example is equivalent to a read-req. It retrieves all attributes and values of one entry:

```
search-req
        base-object =    <countryName "AU">
        <organizationName "Democorp">;
```

Example: Search Service for Single Level

The following example is equivalentg to a list-req. It retrieves the names of all objects from one level under Democorp:

```
search-req
        base-object =    <countryName "AU">
        <organizationName "Democorp">
        one-level-only
        no-info-return;
```

Example: Search Service for Subtree

The following example searches all entries under Democorp, including Democorp. It retrieves all information about every entry that contains an attribute surname with the value *Smith*:

```
search-req
        base-object =    <countryName "AU">
                <organizationName "Democorp">
        whole-subtree
        filter = { attr = surname value = "Smith" };
```

Example: Search Service for Local Directory Tree

Search the local directory tree, retrieving all objects that contain a surname attribute with a value of Smith and a title attribute with a value of *Manager* and a *telephoneNumber* attribute. Retrieve only the *commonName*, *surname*, and *telephoneNumber* attributes of the objects matching the search filter.

```
search-req
        base-object = <>
        whole-subtree
        filter = { and {attr = surname value = "Smith",
                attr = title substrings [ any "Manager"],
                attr = telephoneNumber present}
                }
        attrs = commonName, surname, telephoneNumber
        common-args = { local-scope };
```

search-req-filter

The search-req-filter option defines the filter for the search-req command. The search-req command uses the filter to determine whether to display the entries it finds, or to exclude these entries from its results.

Syntax

A search-req filter consists of one filter item, or a number of filter items linked together by one of the logical operators: *and*, *not, or*.

Hence you can use any of the following formats:

search-filter-item
and {*search-filter-item* ,*search-filter-item* ...}
or {*search-filter-item* ,*search-filter-item* ...}
not {*search-filter-item*}

The braces ({}) in these expressions are part of the syntax and define the scope of the operator.

Wherever you have one search filter item you can replace it with one of the above expressions, and you can can nest these expressions indefinitely.

search-filter-item

Defines an item within a search filter. It has one of the following formats:

`attr=`*`DN`* `present`

`attr=`*`DN`* `value {= | <= | >= | ~=}` *`attributeValue`*

`attr=`*`DN`* `substrings {initial | final | any}` *`attributeValueSubString`*

The braces ({}) and pipe symbols (|) in these expressions are not part of the command—they are there merely to show the alternative forms of the expression: the braces enclose a set of alternatives, and the pipes separate the alternatives.

The operators and other parameters in these expressions are as follows:

=

Equals

<=

Less than or equals

>=

Greater than or equals

~=

Approximately equals

attributeValue

Defines a possible attribute value.

attributeValueSubstring

Defines a possible substring of an attribute value.

LDAP-Only Examples

These commands are supported by the LDUA, which is the LDAP version of the DUA. Because the LDUA uses the LDAP protocol, it has access to LDAP controls, which permit operations to be modified. For example, the search command can include server-side sorting and paged results.

Example: Search and Sort Results

Search all objects under Democorp, retrieving all objects with common name beginning with H, and sort the objects in reverse order by description:

```
search-req
        base-object=<o Democorp>
        whole-subtree
        filter = { attr = commonName substrings [ initial h ] }
        attrs = commonName,description
        controls = { server-side-sort description reverse critical };
```

Example: Search and Page Results

Search all objects under Democorp, retrieving all objects that contain an attribute common name beginning with H, and page the results:

```
search-req base-object=<o Democorp> whole-subtree
        filter = { attr = commonName substrings [ initial h ] }
        attrs = commonName,description
        controls = { simple-paged-results size = 20 critical };
```

Example: Search and Sort and Page Results

These LDUA controls can be used together. The following example searches all objects under Democorp, retrieves all objects that contain an attribute common name beginning with H, and sorts and pages the results:

```
search-req base-object=<o Democorp> whole-subtree
        filter = { attr = commonName substrings [ initial h ] }
        attrs = commonName,description
        controls = { server-side-sort commonName
        simple-paged-results size = 20 position = 20 };
```