CA Directory

Administration Guide

r12.0 SP6

technologies

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 1: About Directories

This section contains the following topics:

What Is a Directory?

A directory lets you store data about people, resources, and systems. Directories are designed to make looking up information as fast and efficient as possible.

An everyday example of a directory is a traditional Yellow Pages business directory. A Yellow Pages directory stores key information about businesses, and is arranged to make it as easy as possible for you to find the business you want.

A Directory Manages Information

A directory is a service for information management. It stores information about people, resources, and systems. A directory allows users to instantly look up critical everyday information.

Example: A Call Center's Customer List

Peter works in a call center, and he receives calls from customers across Asia and Europe. Every time he receives calls from customers, he looks up their details on the customer directory, and each search takes less than a second. This means that after customers have verified their identity, they do not have to tell Peter their account number, address, or any other information that is stored in the directory.

A Directory Can Store Reference Data about Anything

You can use a directory to store any kind of reference data.

Directories use a schema to decide what data is stored about each kind of object. A schema defines the kinds of objects that can be stored in a directory, and the types of information that these objects can contain. You can design your own schema, or you can use an existing schema.

Example: A Manufacturer's Product Catalog

Company A manufactures glass bottles. Their product catalog is linked to a directory that contains data about every bottle they make. The directory lists the height, weight, color, volume, shape, and price of each bottle type. Company A designed a directory schema that includes fields for each of these categories of information.

A Directory Stores Reference Data, Not the Actual Data

Directories are used for fast lookup of key information.

It is usually best to store the actual data being referenced in a database. For example, a directory could be used for storing a music catalog, but not the actual audio data.

A Directory Allows Information to Be Read and Modified

Most users and applications use directories for looking up information. However, depending on permissions, some users can modify the information in the directory. New entries can be added, and existing entries can be modified or deleted.

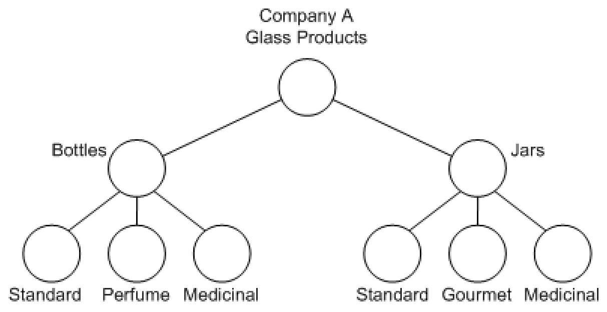
A Directory Can Organize Information in a Hierarchy

A directory can arrange information in a hierarchy, to make it easy to browse. This is useful for lists of products, services, staff, customers that can easily be grouped into distinct categories.

If a directory contains a huge number of entries and it is not browsed by people, you can use a flat structure.

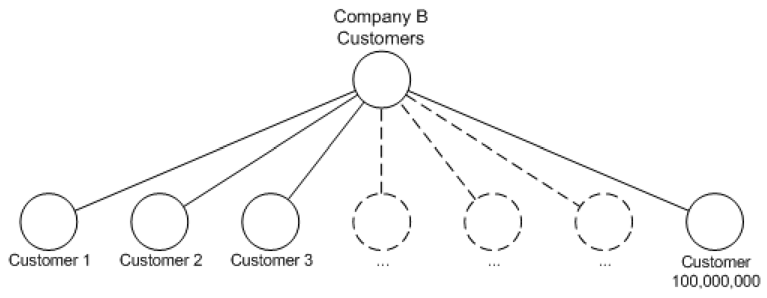
Example: A Hierarchical Directory Structure

Company A produces about four hundred kinds of glass vessels. Company A’s products are easily divided into the following hierarchical categories:



Example: A Flat Directory Structure

For a directory with millions of entries, the following flat structure might work better:



Directories Are Based on Standards

Directories are based on standards. This means that the information stored in a directory can be used by any other system that uses those standards. The two main standards for directories are LDAP and X.500.

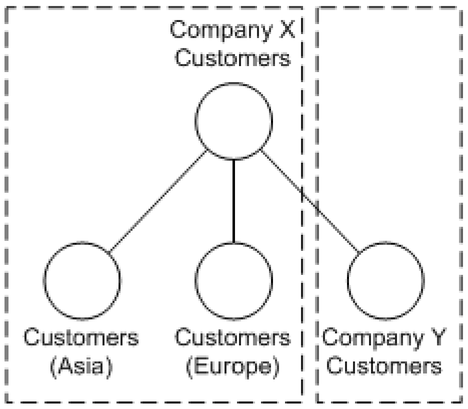
Data in a Directory Can Be Distributed and Replicated

The information in a directory can be divided according to a hierarchy. This means that different sections of the directory can be kept in separate locations, to improve resilience and support replication.

Also, directories can be replicated to multiple locations. This allows for quicker searching by many users in different locations, and it also enhances the robustness of the system. If one computer has a hardware problem, the directory on that computer synchronizes itself with its peers on other computers after the hardware problem is fixed.

Example: Company X Acquires Company Y

The Company X call center recently acquired Company Y, which had data about its customers recorded in a directory. The following diagram shows that the Company X directory has been configured to search this new data, without changing the Company Y directory:



Now, when Peter receives a call from a customer of Company Y, he can use the main Company X directory to look up data about the customer.

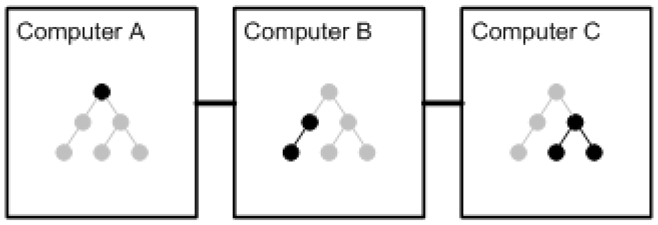
Distribution

Distribution is important for scaling.

Similar to the World Wide Web, distribution lets any number of servers share and maintain their own information. However, unlike the World Wide Web, directory hosts have a server-to-server protocol that enables them to cooperate to provide distributed queries and a unified view of the whole directory information tree.

This means that even though a directory information tree might appear to be a single entity, it may actually be split and stored in separate places.

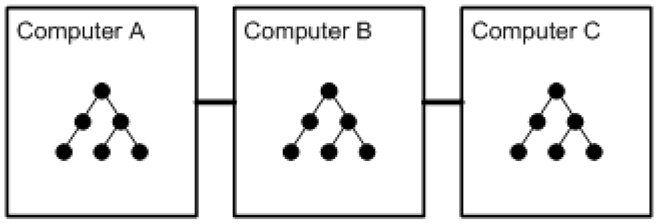
The following diagram shows how one directory service is distributed across three namespaces:



Replication

Replication is important for recovery and sometimes for performance. Replication occurs when the same directory entry namespace exists on different servers. This is similar to the idea of mirror sites on the World Wide Web.

The following diagram shows how one directory namespace is replicated across three servers:



Replication involves copying data. Whenever data is copied, you must ensure that the copies are synchronized. This can make the cost of developing and maintaining replicated systems expensive, and you should weigh it against the benefits of performance and recovery in the event of failure.

CA Directory Uses Both Major Industry Standards

To work together, you have to communicate and share information. This is essential for organizations growing through mergers and acquisitions, and business-to-business communication and eCommerce.

Standards-based directories let this communication take place. However, many directory systems on the market are unable to communicate with other directories.

CA Directory uses both of the following major directory standards:

LDAP

LDAP is a protocol for accessing directories. LDAP is a simplified version of the X.500 directory access protocol (DAP).

X.500

X.500 is a set of computer networking standards that define directory services. The protocols defined by the X.500 standards include DAP, DSP, and DISP. A directory that follows the X.500 standard has distributed operations, distributed management, distributed security, and replication.

LDAP is important for clients, while X.500 is important for servers.

CA Directory fully applies X.500 and LDAP standards to provide a distributed and reliable directory service. CA Directory uses LDAP support to access LDAP-only directories, and the X.500 distributed directory model for distribution.

In addition to supporting LDAP for access, CA Directory permits the integration of LDAP-only servers to a directory backbone.

Chapter 2: How CA Directory Works

This section contains the following topics:

DSAs

A *DSA* is a process that manages some or all of a directory's namespace.

Types of DSA

Router DSA

A *router DSA* has no local data and no datastore. It can only route traffic to other DSAs.

DXmanager represents a router DSA with this icon:

Data DSA

A *data DSA* holds data, and queries are routed to it by router DSAs. A data DSA can perform local operations, replicate updates, and route traffic to other DSAs.

A data DSA is always associated with a datastore, which stores the data even if the DSA is shut down.

The datastore is a memory-mapped file. The DSA uses the in-memory copy of the namespace partition, and relies on the operating system memory mapping functions to keep the file copy on disk up to date.

DXmanager represents a data DSA with this icon:

Third-party X.500 DSA / Third-party LDAP Server

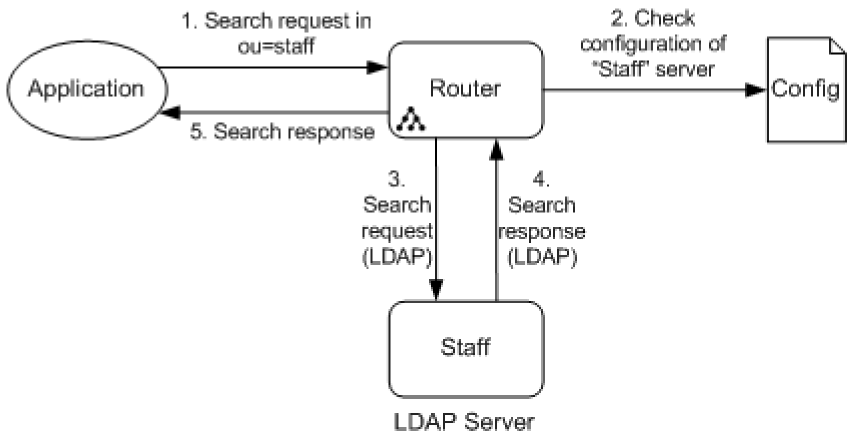
This is a DSA from an external vendor.

DXmanager can monitor these DSAs if they are suitably configured to accept requests from the backbone.

DXmanager represents a foreign DSA with this icon:

Example: How a Query to an LDAP Server Works

The following diagram shows how a CA Directory DSA deals with a query destined for an LDAP server:


1. Search request in ou=staff
2. Check configuration of "Staff" server
Application
Router
Config
5. Search response
3. Search request (LDAP)
4. Search response (LDAP)
Staff
LDAP Server

1. The router DSA receives a search request.

   The search specifies a branch in the Staff partition, which the router DSA knows is served by the LDAP server Staff.

2. The router DSA checks the Staff server's configuration to see whether there are any special conditions for linking to that DSA. The configuration indicates that the Staff server requires that any links use LDAP.

3. The router DSA uses acts as a client to make an LDAP search request of the Staff server.

4. The Staff server responds with the search result (also using LDAP).

5. The router DSA returns the search result to the client.

How to Create a Data DSA

You can create a data DSA with DXmanager or by using the DXnewdsa tool. The DXnewdsa tool provides, in an initialization file, the core commands that are needed to create a DSA. You can also create these files yourself. The steps needed to to create a DSA are as follows:

1. Set the datastore location (see page 27)
2. Create the datastore (see page 27)
3. Set up the knowledge and schema configuration (see page 33).
4. Index the attributes used in search filters (see page 27).
5. Enable the use of the memory-mapped datastore (see page 27).

Set the Datastore Location

You need to set the datastore location. To do this, include the following command in the DSA initialization file:

```
set dxgrid-db-location = datastore-fullpathname;
```

Create the Datastore

You need to create a datastore for each DSA. To create a datastore, use the DXnewdb tool. See the *Reference Guide* for details on how to use the DXnewdb tool.

The DXnewdsa tool uses DXnewdb when it creates a new DSA.

Index the Attributes Used in Search Filters

You must index the attributes that are included in search filters.

Usually you index all attributes. Only restrict the list of indexed attributes if you want to save space.

To index attributes, add the following command to the initialization file:

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

Note: this command must occur after the schema has been defined.

Example: Index All Attributes

The following command indexes all attributes

```
set cache-index  = all-attributes;
```

Enable the Use of a Memory-Mapped Datastore

A data DSA needs a memory-mapped datastore. To enable a DSA to use a memory-mapped datastore, add the following command to the initialization file:

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

Ensure that this command appears after any other commands in the initialization file. The DSA loads the memory-mapped file as soon as it reads the *set lookup-cache* command.

Using DXmanager to Create a DSA

If you use DXmanager to create a DSA, you need to deploy the configuration.

Deploying is the process of transferring the XML configuration file from DXmanager to the DSA's host, and then reinitializing the DSAs.

When DXmanager creates a DSA, it also creates a default DSA initialization (.dxi) file. Edit this file to ensure that all of the required configuration files are sourced.

If you edit and deploy the configuration again, the initialization files are not overwritten. This means that you do not need to change the initialization files again - your customizations remain in the files.

When DXmanager removes a DSA, it gives the .dxi file an extension of .old.

How DXmanager Names DSAs

When you create a DSA using DXmanager, it forms the DSA name according to the following convention:

```
namespace-host
```

namespace is the name of the namespace partition.

host is the name of the host.

For example, a data DSA on the host named *Server15* that serves the namespace partition of *Staff* is given the following name:

```
Staff-Server15
```

Note: When you create a DSA outside of DXmanager, avoid the DXmanager naming convention. Using a different naming convention allows DXmanager and non-DXmanager DSAs to exist side by side without confusion.

More information:

DSA Initialization Files (see page 35)

Advanced Indexing Commands

Reverse-Index Attributes

For some attributes, such as phone numbers, the only significant differences in the values occur towards the end of the value. For example, all users might have the same leading digits in their telephone numbers. In such cases, it makes searching faster if you reverse index the attribute, so that the index is created on the values after reversing the order of the bytes.

CA Directory does not use a reverse index in preference to a regular forward index. It is only used in substring lookups where you use specify final as an allowed filter.

To reverse-index an attribute, add the following command to the initialization file (.dxi):

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

For example, 123-4567 is indexed as if it were 7654-321.

Increase the Maximum Number of Indexed Entries in a DSA

To increase the number of entries in a DSA, insert the following command in the initialization file:

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

This command sets the maximum number of entries (or distinct attribute values for each attribute type) that a DSA can index.

If you leave this setting at its default value, a DSA can contain up to about 8 million entries, or about 8 million distinct attribute values. For example, if each entry had two unique telephone numbers, then by default the cache could hold only 4 million entries. Attributes that are not indexed are not affected.

You should only use this command if you are sure you need more than 8 million entries or distinct attribute values. If you set this value higher, the DSA will have bigger memory requirements, and its performance may degrade.

Configuration

You can configure CA Directory using commands in text files. The text files contain commands that define how the DSA works. These commands are identical to the commands that can be entered from a DSA console.

Note: If you have DXmanager installed, use it to define the knowledge. It stores the knowledge in an XML file. Do not edit this file yourself: use DXmanager instead.

DSA Knowledge, DXmanager and the set dsa Command

The information a DSA needs about other DSAs (and non-DSA directories): how to connect to them, their role (for example, replicas) and the entry namespace they manage is referred to as knowledge. The total knowledge of the system is used to form a backbone of directory servers to which each DSA belongs.

It is important that all DSAs in a backbone have a consistent view of each other. For this reason it is standard practice to ensure that all DSAs share the same knowledge.

There are two mechanisms you can use to define the knowledge: DXmanager and the set dsa command. They are mutually exclusive: If you have DXmanager installed, DSAs do not allow the set dsa command: if the command is entered at the console the DSA returns a syntax error, and if the command appears in a configuration file, it stops the DSA.

If you have DXmanager installed, use it to define the knowledge. It stores the knowledge in an XML file.

If you do not have DXmanager installed use the *set dsa* command to define the knowledge. The DSA knowledge is stored in a configuration file (.dxc) in the knowledge directory (along with other DSA definitions). A DXserver determines its own entry (identity) by looking for its own name among all the set dsa definitions.

Note: A DSA must use its own name (case insensitive) from the initialization file: *serverName.dxi*.

Note: The commands in the knowledge file must appear in the correct order. For information about this order, see the text file DXHOME\config\knowledge\knowledge.help.

The DSA Settings that DXmanager Manages

DXmanager can manage all the settings you can change with the set DSA command.

You now change the following settings in DXmanager. Most settings are changed either on the namespace partition screens or on the topology screens.

- address interface—A topology (host) setting
- address port—A namespace partition setting
- allow-check-password trust flag—A namespace partition setting
- allow-downgrading trust flag—A namespace partition setting
- allow-upgrading trust flag—A namespace partition setting
- auth-levels—A namespace partition setting
- console-port—A namespace partition setting
- console-password—A namespace partition setting
- disp-psap—A namespace partition setting
- dsa-password—A namespace partition setting
- dsp-idle-time—A namespace partition setting
- dsp-ldap link flag—An external DSA setting only
- dsp-ldap2 link flag—An external DSA setting only
- dsp-ldap-proxy link flag—An external DSA setting only
- ldap-dsa-name—An external DSA setting only
- ldap-dsa-password—An external DSA setting only
- limit-list—A DSA setting only
- limit-search—A DSA setting only
- ms-ad link flag—An external DSA setting only
- ms-exchange link flag—An external DSA setting only
- multi-write-disp-recovery—A namespace partition setting
- multi-write-queue—A namespace partition setting
- multi-write-serial—An external DSA setting only
- native-prefix—An external DSA setting only
- no-routing-ac—A namespace partition setting
- no-server-credentials trust flag—An external DSA setting only
- no-service-while-recovering—A namespace partition setting
- osi-psap—A namespace partition setting

- prefix—A namespace partition setting
- read-only—A namespace partition setting
- relay—A DSA setting only
- remote-console-port—A namespace partition setting
- snmp-port—A namespace partition setting
- ssl-encryption link flag—A namespace partition setting
- ssl-encryption-remote link flag—A namespace partition setting
- trust-conveyed-originator trust flag—A namespace partition setting
- unavailable link flag—A DSA setting only
- wait-for-multiwrite—A namespace partition setting

DXmanager now sets the following settings automatically, and you cannot directly change them:

- always-chain-down—The DSA always chains down.
- ca_ldap
- check-roles-with-dsa-credentials—Superseded by the trust DSA-triggered operations, which are set using DXmanager.
- check-unique-attrs-with-dsa-credentials—Superseded by trust DSA-triggered operations, which are set using DXmanager.
- dbconnection—The DSA issues a cautionary alarm and ignores the user and password items.
- dsa-name—DXmanager sets this automatically .
- load-share—Always enabled for data DSAs.
- load-share-group—DXmanager sets this automatically to the site name.
- min-auth—DXmanager sets this to the minimum authentication level of all DSAs in the grid.
- multi-casting—DXmanager sets this automatically to TRUE.
- multi-chaining—DXmanager sets this automatically to TRUE.
- multi-write
- multi-write-async
- multi-write-group—DXmanager sets this automatically to the region name.
- precedence—DXmanager sets the DSA precedence by current region first.
- servername
- shadow
- write-precedence—DXmanager sets the DSA precedence by current region first.

Configuration Files

The configuration files are stored in the subdirectories under *DXHOME/config*, as follows:

access

Contains access control commands. See Set Up Access Controls (see page 153).

Usual location: DXHOME/config/access/

knowledge

Contains the knowledge. The knowledge defines the DSA.

The knowledge configuration file contains just the set dsa command, but that command defines many settings.

The core knowledge for a DSA is the DSA's prefix, address and port.

Knowledge is shared among DSAs. A DSA needs to see the knowledge of any other DSA that it connects to.

We recommend that you allow all DSAs to see all other DSAs' knowledge.

Note: If you use DXmanager, this file is replaced by an XML file. Do not edit this XML file yourself. DXmanager manages it.

limits

Contains commands that limit the behavior of the DSA. See Limits That You Can Change.

Usual location: DXHOME/config/limits/

logging

Contains commands that define which logs are in use, and sets the level of information contained in each type of log. See Logs (see page 75).

Usual location: DXHOME/config/logging/

replication

Contains commands that define DISP replication. See Set Up DISP Replication (see page 112).

Usual location: DXHOME/config/replication/

Note: Multiwrite and Multiwrite DISP do not use these commands.

schema

Contains commands that formally define the contents and structure of the directory data. See Set Up Schemas (see page 77).

Usual location: DXHOME/config/schema/

server initialization

Contains the commands that initialize a DS and calls all other settings. See DSA Initialization Files (see page 35).

Usual location: DXHOME/config/servers/

settings

Contains the remaining commands that define how the DSA operates.

Usual location: DXHOME/config/settings/

SSL certificates

Contains SSL certificates.

Usual location: DXHOME/config/access/

Configuration Files and Filename Extensions

The different types of CA Directory configuration files are as follows. The file extensions shown here are a convention to help recognition, not mandatory.

File Type	Extension	How Many?	What the Files Do	How the Files Are Edited
DXmanager configuration file	.xml	One	Contains the topology and namespace partitions of the whole backbone, plus some configuration of each DSA. DXmanager uses this instead of knowledge files. This information is shared by all DSAs that are configured thorugh DXmanager	This file is created by DXmanager. Never edit this file yourself: use DXmanager instead.
DSA Initialization files	.dxi	One for each DSA	Contain commands to source .dxg and .dxc files. When a DSA starts, it uses an initialization file named *dsaname*.dxi.	In a text editor
DSA configuration files	.dxc	As many as required	Contain one or more commands that set configuration parameters.	In a text editor
Group files	.dxg	As many as required	Contain a series of source commands to group one or more .dxc files in the current directory.	In a text editor
Script files	.dxs	As many as required	Contain commands supported by a DSA console. You usually use script files for testing.	In a text editor

DSA Initialization Files

Every DSA has an initialization file, which is usually in the *DXHOME/config/servers* directory. The name of the initialization file must be the same as the DSA name.

Each DSA's initialization file includes source commands to refer to files in other config subdirectories.

More information:

How DXmanager Names DSAs (see page 28)

How a DSA Uses Its Initialization File

When an administrator starts a DSA, the DSA uses its initialization file in the following way:

1. The DSA looks for a file in *DXHOME/config/servers* that has the same name as the DSA and a .dxi extension.

   For example, a DSA named *Democorp* looks for the file *Democorp.dxi*.

2. The DSA reads the initialization file and executes the commands in the order they are read.

   A source command causes the DSA to read the sourced file, and execute commands in that file.

The order of commands and sourced files is important because some commands require certain settings to be defined. For example, access control rules require that attributes are already defined, so the schema (which defines attributes) must occur before access control rules are defined.

If an error occurs while reading the configuration files, the DSA cannot start. You can find information about this error in both the operating system error log and the DSA alarm and trace log files in the *dxserver/logs* directory.

Schema Files

The schema of the CA Directory server is in its schema directory (for example, DXHOME/config/schema), and consists of individual schema configuration files (.dxc) and group files (.dxg).

The schema of a running directory is available to LDAP clients through the LDAP Version 3 mechanism of schema publishing. LDIF is a convenient format in which to publish the schema.

To use new schema, it must be made available to the directory and the DXtools. This means a new or updated .dxc file, and possibly an updated .dxg file.

Script Files

Script files store frequently used, or complex commands (such as a series of searches). You can execute these files from a DSA console, or by using the source command, from other script files.

Script files can the following sorts of commands:

- Internal commands, such as *source*
- Configuration commands, such as *set trace*
- DUA commands, such as *bind-req*

See the Reference Guide for more details.

Group Files

A configuration directory for a component can contain a number of configuration (.dxc) files. For complex configurations with many DSAs, you might need many configuration files, grouped in a convenient manner using group (.dxg) files.

Reasons for grouping files include the following:

- To share the same set of configuration files among many DSAs
- To enable a .dxi file to source a single configuration group (.dxg) file for one configuration component (such as limits).
- To manage ordering of the configuration files. This is especially important in schema where definitions in one file depend on definitions in another file.

Example: Using a Group File for Schemas

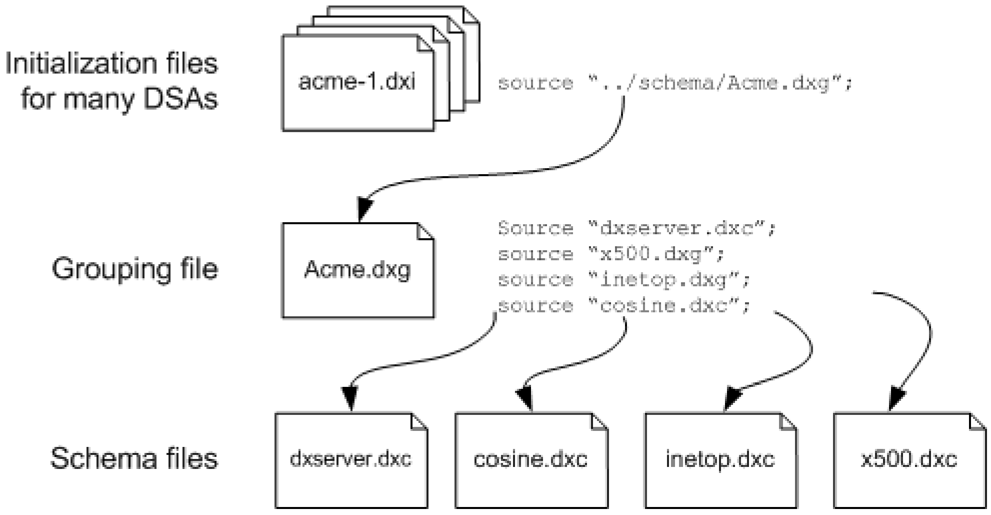
The Acme Company has a directory that uses a number of DSAs. Each DSA uses the same schema files.

Instead of each DSA initialization file sourcing the schema files directly, the directory administrator uses a group file named Acme.dxg, which sources the schema files. Each DSA initialization file sources Acme.dxg, by including the following line:

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

If the DSAs require additional schema, the administrator only needs to edit one file, Acme.dxg.

The following diagram shows the configuration files and the source statements:



View the Configuration

To view the memory configuration variables and their values, read the configuration files, use DXmanager, or use the get cache console command.

You can also put the get cache command in the initialization file to print the configuration to the log on startup. In that case, the command must be after the lookup-cache command, for example, as follows:

```
set lookup-cache = true;
get cache;
```

IP Set Up

CA Directory works across an IP network and requires that the following IP related features are in place:

Firewall access for ports used for multiwrite

Any firewall between hosts must allow bidirectional access to the ports used for DSAs to send and receive multiwrite access.

If you run DXmanager, be aware of the following IP requirements.

Static and dynamic IP addresses

Because DXmanager refers to DSA hosts by their DNS name, the hosts can have dynamic IP addresses.

DXmanager host must have a static IP address.

Firewall access for ports used by DXmanager and dxadmind

Any firewall between the DXmanager host and a DSA host must allow bidirectional access to the ports that DXadmind uses. By default this is port 2123.

Host names resolution to network wide IP addresses

On UNIX machines some host files have an entry that gives the host name of a machine as a local address. For example if the host name is host1, then the entry might appear as:

```
127.0.0.1 host1
```

This causes the DSA to fail. Provide the global IP address of the host or use DNS to resolve the host name.

Reverse DNS if you use host names

If you specify the DXmanager host to a DSA host in terms of its host name, then you need to have reverse DNS enabled. When the DXmanager sends a message, DXadmind needs reverse DNS to match the message's source IP address to the host name that you provided.

Configuring Alternative Network Addresses

You can configure more than one network address for a DSA. Do this when you want to distribute the directory over more than one physical or logical network. This facilitates increased network availability by using additional networks between DSAs.

To configure more than one network address for a DSA, use the Connections area in DXmanager.

If your firewall translates remote addresses, your DSA may not recognize the remote DSA because of the changed address. You must configure alternative network addresses in the knowledge file of the remote DSA by specifying the address of the remote DSA followed by the address of the firewall.

The Backbone Concept in DXmanager

The *backbone* is the term used in DXmanager to refer to the directory installation as a whole. It includes all the DSAs and their configuration data.

DXmanager represents the backbone using this icon:

All the DSAs in a backbone combine together to service one namespace. Because the namespace is a hierarchical tree it can be divided into partitions. A DSA manages one namespace partition. On small installations a DSA might manage a whole namespace.

When you use DXmanager you set properties in the backbone by editing the properties of one of the following:

- The namespace
- The network hardware and software that together runs the directory.

DXmanager applies the relevant properties of the namespace and of the network to the DSAs that it manages.

Network Topology in DXmanager Terms

For DXmanager, a Network topology consists of the following:

- Hosts (computers)
- Sites (collections of hosts)
- Regions (collections of sites)

Hosts are contained in sites, and sites are contained in regions.

When you configure a setting for an item, it becomes the default for all the items contained within it. For example, a setting defined for a region becomes the default setting for all the sites in that region (and therefore for all hosts in those sites).

DXmanager lets you define the network topology of the systems in which the directory runs.

Hosts

A *host* is a single computer with CA Directory installed on it. A single host may serve one or more namespace partitions.

Note: The *directory management server* is the computer on which you have installed the Directory Management package, which includes DXmanager. Do not define the directory management server as a host to DXmanager unless it also serves a namespace partition.

DXmanager represents a host with this icon:

Sites

A *site* is a DXmanager term for collection of hosts that are connected by a reliable, fast and low latency network, such as a LAN. hosts A site corresponds to a load sharing group.

We recommend that you equate sites to your actual data centers.

Each host belongs to exactly one site.

DXmanager represents a site with this icon:

If a number of hosts within a site manage the same namespace then DXmanager shares the load between the hosts.

A router does not send read queries to a different site if there is a DSA in the same site that can handle the query.

If there is no DSA on the same site that can handle a read query, the router sends the query to another site in the same region, and if there is no DSA available there, it sends the query to another region.

The router sends updates to only one DSA in each region. It uses the first DSA in its list of DSAs.

Regions

A *region* is a collection of sites. Multiwrite replication between DSAs in the same region is synchronous.

Each site belongs to exactly one region.

You should define multiple regions to separate sites that have poor-quality connections. Doing this helps minimize the performance cost of replicating updates across these links. However, use just one region to cover for hosts that are connected by a fast network.

Multiwrite replication is asynchronous between regions. This means that the poor quality of the connections does not unduly slow directory performance.

Similarly, because regions are connected by slow links, CA Directory prevents load-sharing between regions unless load-sharing within a region is not possible.

DXmanager represents a region using this icon:

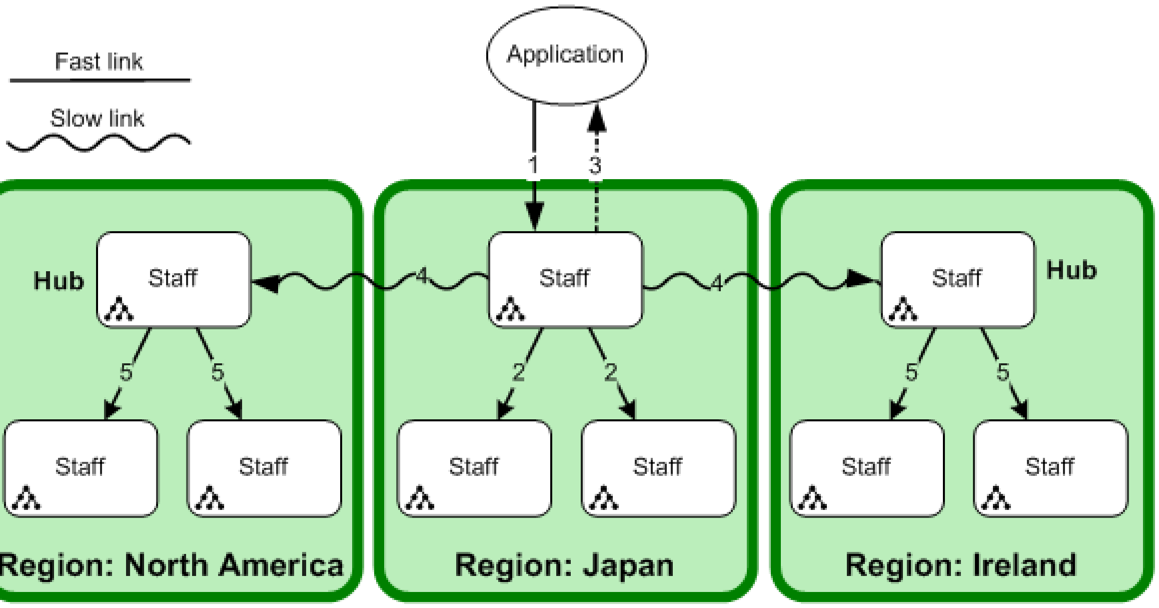
Regions Avoid Replication Bottlenecks

If your backbone includes any slow network connections, you should set up regions. DSAs connected by slow links should be part of different regions.

For each namespace partition, each region has one hub DSA. This is the DSA that accepts multiwrite requests from DSAs in other regions.

Example: Multiwrite System with Regions to Avoid Bottlenecks

The following diagram shows a backbone with one namespace partition (Staff) across three regions:



The diagram shows the following steps:

1. **Write to self:** A client sends an update request to a DSA, which applies the update to itself.

2. **Write to peers in region:** If the local update succeeds, the DSA sends the request to its peers in the same region. If these updates succeed, the peers send confirmations back to the first DSA.

3. **Send response to client:** When the first DSA has received confirmation from each peer in its region, it sends the confirmation response to the client.

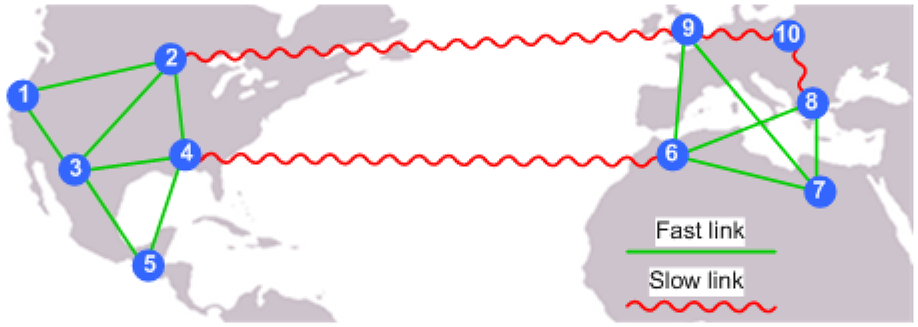
4. **Write to hub DSAs in other regions:** The first DSA sends the request to the hub DSAs in each of the other regions.
5. **Hub DSAs write to peers:** Each hub DSA sends the request to the other DSAs in its region.

   The following steps are not shown in the diagram.
6. **Peer DSAs write to self:** Each peer DSA makes the update.
7. **Peer DSAs send confirmation to hubs:** Each peer DSA sends confirmation of the update to the hub DSA.
8. **Hub DSAs send confirmation to the first DSA:** Each hub DSA sends the confirmation response to the first DSA. This DSA has already sent confirmation back to the client, so the client is not affected by the slow links.

Example: A Backbone with Three Regions

A global company has directory hosts in North America, Central America, North Africa and Europe.

This diagram shows the speed of the network connections between the sites:



Sites 1, 2, 3, 4, and 5 are linked by fast connections. Sites 6, 7, 8, and 9 are also linked by fast connections.

However, these two groups of sites are connected by slow links. Also, Site 10 is only connected to other sites by slow links.

The directory designers decide to create the following regions and sites in DXmanager:

Region A	**Region B**	**Region C**
Site 1, Site 2, Site 3, Site 4, Site 5	Site 6, Site 7, Site 8, Site 9	Site 10

Configuration Inheritance in DXmanager

DXmanager applies hierarchical inheritance to the configuration of CA Directory. You can set values for configuration items at the following levels, which are listed here in order from the broadest (least specific) level to the most specific:

1. **Backbone—(broadest level)** Works as a default value for the whole directory.
2. **Namespace partition—**This affects all DSAs that serve that partition.
3. **Region—**This affects all DSAs in that region.
4. **Site—**This affects all DSAs in that site.
5. **Host—**This affects all DSAs on that host.
6. **DSA—(Most specific level)**

Using DXmanager, you can create configuration values at any of these levels A value at a more specific level overrrides a value at a broader level. So, for example, a value specified at DSA level overrides a value specified at Namespace partition level.

When you change a configuration value, all inherited instances of that configuration item change as well.

DXmanager displays which configuration values are inherited and where they are inherited from.

Not all configuration items can be set at all levels. In particular, you cannot set values in the network topology levels for configuration items that are applicable to namespaces.

We recommend that you set configuration values at the broadest level possible. This makes it easier for you to make changes across the backbone, also means that you do not need to specify so much new information when you add a new item.

Example: Setting the SSL Port Number

The SSL port number defines the port that a DSA uses to communicate with its SSL daemon.

To set this up, you could define this port number at the host level. However, this would mean that you would have to set this for every host. Instead, we recommend that you set the SSL port number at the backbone level. This then works as a default value for the entire directory backbone.

If the port number you choose is already in use on a particular host, you can set a different value for that host, which overrides the value you set for the entire backbone.

Namespace Partitions

A *namespace* is the tree of all data in the directory, synonymous with directory information tree. It is defined by the DN of the top node in the tree.

A *namespace partition* is a sub-section of the directory information tree. (The partition might cover the whole namespace.)

Each DSA serves a namespace partition. When a namespace is partitioned to be served by more than one DSA, the directory is said to be *distributed*. A well distributed directory provides good scalability and performance.

You can control each namespace partition separately. Each partition can have different security and can be located on a different set of hosts than other namespace partitions.

If more than one DSA serves the same namespace partition, the partition is said to be *replicated.*

DXmanager represents namespace partitions using the following icons:

- Replicated namespace:
- Unreplicated namespace:

Reasons to Partition a Namespace

Consider dividing a namespace into partitions if any of the following are true:

The size (memory usage) of a namespace is too big for one DSA

Partitioning the namespace amongst DSAs is the only effective way to deal with this problem.

One subtree has a much higher number of operations than other subtrees

Placing the subtree in its own partition means DSAs can be dedicated to serving just the busy subtree, and queries for other areas of the namespace are served by other DSAs. This usually improves performance

A subtree is particular to a specific business function or geographical area

This lets you maintain and control only the DSAs that serve that subtree, set up different security for those DSAs, or locate that subtree on a different set of hosts than the other areas in the namespace.

How DXmanager Uses the Topology to Configure the Backbone

When you define the network topology, DXmanager automatically uses this information to set up replication, failover, and load-sharing.

Regions and Replication

DXmanager automatically configures multiwrite replication between DSAs that serve the same namespace partition, by putting them in the same multiwrite group.

Replication within a region

This occurs synchronously. When a DSA receives an update request, it replicates it synchronously to all other DSAs in the same region.

Replication between regions

This occurs asynchronously. When a DSA receives an update request, it replicates it asynchronously to one DSA in each other region. The receiving DSAs then forward the request to other DSAs in the same region.

This 'send and forward' model reduces the network traffic between refgions.

For the application that sent the update request, this means that it waits only until the DSAs in the same region have applied the update.

Note: DSAs in different regions might be briefly inconsistent with each other.

More information:

What Is Replication? (see page 95)

Sites and Load Sharing

Load sharing lets a router DSA distribute incoming requests evenly among all DSAs in the same site that serve the same namespace partition. This improves performance.

DXmanager automatically sets up load sharing between DSAs in the same site that serve the same namespace partition. If you do not use DXmanager you need to set the DSA knowledge to enable load sharing.

More information:

Load Sharing (see page 125)

Sites and Failover

Failover is the ability of a router DSA to continue to service queries even when a data DSA becomes unavailable. If the router detects that a DSA has failed, it resends outstanding requests to another DSA that serves the same partition, making the failure invisible to clients.

If no DSAs within the router DSA's site are reachable, the router DSA sends queries to a DSA in another site in the same region.

If a router cannot find a suitable DSA in any site in the same region, then it sends the query to another region. If this happens, the performance is probably significantly lower, but the service is maintained.

Note: Failover is managed by a router, but it requires that the DSAs in the group maintain their synchronicity, which they do by using replication.

More information:

Failover and Failback (see page 126)

Assigning the Topology Components to the Namespace Partitions - Instantiation

In DXmanager, the process of assigning hosts, sites, or regions to namespace partitions is called *instantiation.* After the configuration is deployed, this process creates one or more DSAs to serve the namespace partitions.

At the end of this process, each namespace partition has one or more DSAs assigned to it.

DXmanager does not do everything required to create a DSA. You still have to provide any DSA specific properties. You also need to do the following tasks:

- Set up any non-DXmanager configuration; that is, all the non-knowledge configuration. There is also some knowledge configuration that is not yet set up by DXmanager
- Load or synchronize data into the datastore

How DXmanager Works with Third-Party LDAP Servers

You can enter details of LDAP servers into DXmanager, and they appear on the maps, and DXmanager enables the DSAs to talk to the LDAP servers. However, DXmanager cannot currently monitor or control these LDAP servers.

Chapter 3: Tools to Manage CA Directory

CA Directory includes the following tools to help you set up, control, and monitor your directory backbone:

DXmanager (see page 50)

DXmanager is a web application that lets you create, configure, monitor, and control your directory backbone.

DSA console (see page 65)

The *DSA console* lets you connect to a DSA to give DXserver commands, receive trace information, and act as a user agent.

JXweb (see page 72)

JXweb is a general purpose LDAP-compliant directory browser and editor, which lets you access a directory through the Web.

DXtools (see page 75)

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.

Logs (see page 75)

The *logs* contain output from a DSA, including its trace, diagnostics, warnings, and alarms.

Traces (see page 75)

A DSA's *trace* is its record of almost all operations going into and out of that DSA. You can view a DSA's trace in the log files, and also by using the DSA console.

Alarms (see page 76)

Alarms are reports of critical events that should be monitored.

DXmanager

DXmanager is a web application that lets you create, configure, monitor, and control your directory backbone.

For information on how DXmanager works, see the following:

- How DXmanager Starts (see page 51)
- How DXmanager uses DXadmind (see page 52)
- How DXmanager is secured (see page 53)
- Troubleshooting DXmanager (see page 60)

For information about the main components managed by DXmanager, see the following:

- Backbone Components (see page 40)
- Namespace Partitions (see page 46)
- DSAs (see page 25)

For information about using DXmanager to monitor your directory backbone, see the following:

- Dashboard (see page 251)
- Maps (see page 252)
- Charts (see page 255)
- Alerts (see page 256)

For information about other DXmanager screens, see the following section:

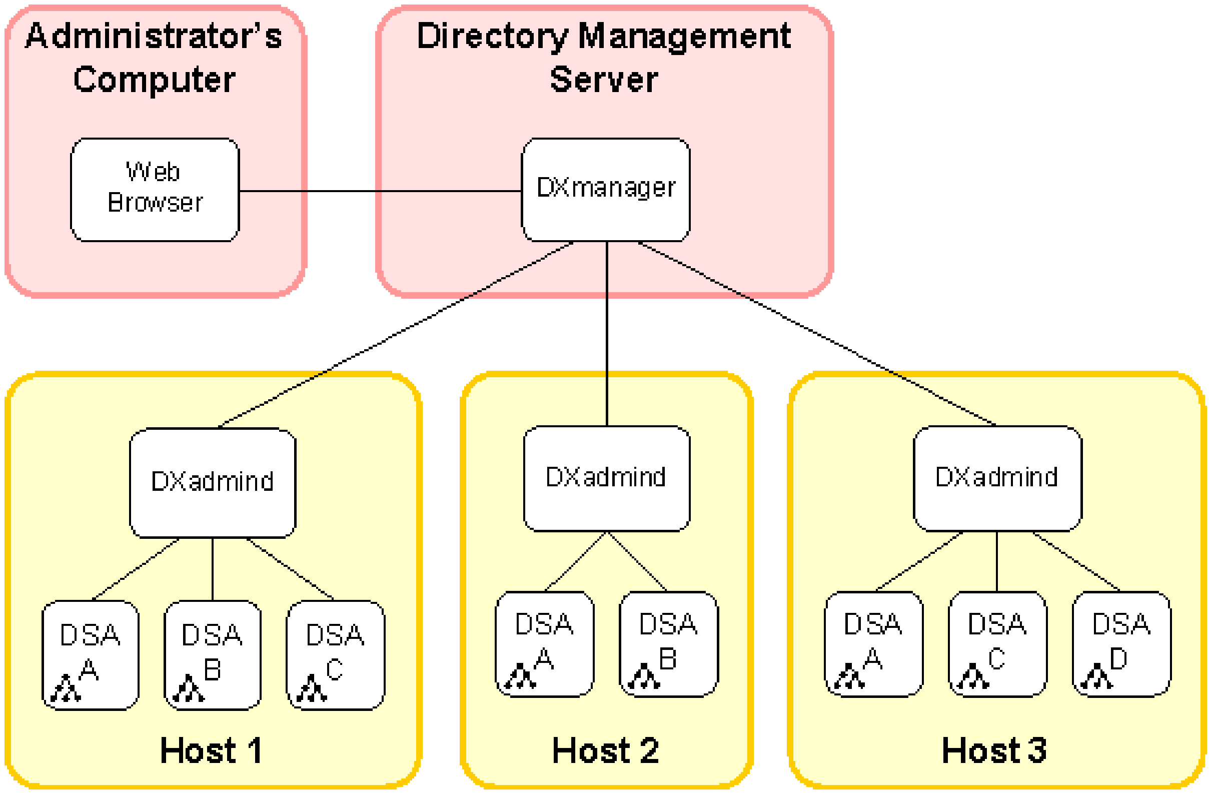
- DXmanager Screens

How DXmanager Starts

DXmanager is usually installed on a server, and is accessed using a web browser from the administrator's computer.

Note: For more information about which computers should be set up, see the *Installation Guide*.

The following diagram shows the components that DXmanager connects to.



The following steps describe how DXmanager starts, as shown in the diagram:

1. A directory administrator uses a web browser to connect to DXmanager.
2. The directory administrator logs in to DXmanager.
3. DXmanager sends the user credentials to the Tomcat realm for authentication.
4. Tomcat checks the supplied credentials against the configured login account repository (an XML file or an an external directory or database) to:
   a. See if the administrator is allowed to access DXmanager
   b. If they are allowed access, at what level.
5. Tomcat provides DXmanager with the administrator's access level.
6. DXmanager displays the information that the administrator is entitled to see.

Depending on their access level, the administrator can now use DXmanager to do the following tasks:

- Monitor the backbone.
- Start, stop, and reinitialize any DSAs in the backbone.
- Change and deploy the namespace partitions and the topology (hosts, sites, and regions).
- Add or remove DXmanager administrators and change their roles.
- Reset the passwords of DXmanager administrators.

How DXmanager Uses DXadmind

DXadmind is installed on each host as part of the Directory package. *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.

This means that DXmanager must have access to the DXadmind on every directory host, and each DXadmind must trust DXmanager.

DXadmind collects the following information and sends it to DXmanager:

- SNMP statistics and traps for each DSA on the host
- The state of each DSA on the host (running, stopping, stopped, or unknown)
- The version of the configuration file on the host
- Whether the configuration file on the host has been tampered with

DXadmind can take the following actions when DXmanager requests them:

- Control the DSAs on the host (starting, stopping, and reinitializing)
- Deploy the configuration from DXmanager, and update the hosts' configuration to match the new configuration from DXmanager

How DXmanager Is Secured

DXmanager has the following security:

- DXmanager accepts only encrypted HTTP connections.

  This prevents the connection between DXmanager and the browser from being intercepted.

- All users must log in to access DXmanager, which uses Tomcat realm based authentication and authorization.

  This ensures that only people with sufficient permission can view, control, or configure a directory backbone.

- DXmanager uses certificate-signed Java applets.

  When you first open DXmanager, the browser asks if you trust the certificate authority that has issued the certificates.

  After you agree to this, DXmanager opens.

Login Accounts

To use DXmanager, users must log in. You need to create a login account for every user who needs to use the product, including the following:

- DXmanager super users
- DXmanager configuration editors
- DXmanager controllers
- DXmanager monitors

Note: When you install DXmanager, the installation creates some sample accounts. You should customize these accounts to reflect your own user accounts.

Methods for Setting Up Login Accounts

DXmanager uses a Tomcat web server, also known as CA Directory Web Server for realm based authentication and authorization. You need to configure Tomcat with authentication details for the login accounts you require.

Note: Do not use this instance of Tomcat to host any other applications. It is customized to work with DXmanager only.

You can create login accounts in these ways:

- Link Tomcat to an existing directory or database of accounts

  If you link Tomcat to a directory or database that contains your organization's user accounts, Tomcat can use the details in this repository as login accounts for DXmanager.

  We recommend you use this method for a production system

  Important! Using this method, you need to restart the Directory Management Server once, after you set up the connection to the directory or database.

- Link Tomcat to an XML file containing account details

  This is the default method. The sample login accounts that are set up during installation are created using this method.

  Although this method is simple, you need to restart Tomcat after adding or editing an account and it is inconvenient to add large numbers of accounts to the file.

  We recommend that you do not use this method for a production system. You can use it for a small test system.

Authentication Using Existing Accounts

If you already store account and role data in a database or directory, you can set up DXmanager to use this data for its own authentication.

If you use a database, it must contain at least two tables, including the following:

- A table containing one record for each account.

  This table must contain a column of account names and one of passwords.

- A table containing one record for every role that is assigned to the accounts.

  This table must contain a column of account names (the same values as in the previous table) and one of roles.

If you use a directory, each required login account must have in the directory a unique entry that corresponds to an element in the initial DirContext defined by the connectionURL attribute.

Note: For more information about the structure of the directory, check the Tomcat realms on the Apache site.

Configure Authentication to a Directory or Database

If you want to use existing user data for DXmanager authentication, you can set up a Tomcat realm. A realm is a store of user names, passwords, and roles that identifies valid users.

Note: Before you begin, stop the Tomcat service: CA Directory Webserver. This disconnects any currently open sessions.

To configure authentication to use an existing directory or database

1. (Optional) If your directory or database includes hashed passwords, enable hashed passwords in Tomcat (see page 58).
2. Download the JDBC or JNDI driver for your database or directory, and put it in the following location:

   ```
   DXHOME/../dxwebserver/lib
   ```

3. Create a new user account for the directory or database, or identify a suitable existing account.

   This account must have at least read-only access to the user and role data. Tomcat connects to the directory or database using this account.

4. Identify the structure of the user and role data in the directory or database, using the instructions in Tomcat realms on the Apache site.
5. Open the following file in a text editor:

   ```
   DXHOME/../dxwebserver/conf/server.xml
   ```

6. Find the following element and comment it out:

   ```
   <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
   resourceName="UserDatabase"/>
   ```

   This disables the default realm, which uses accounts in a local XML file for authentication.

7. Define a realm element for the type of directory or database that you plan to use.

   Some realm elements are already included in the XML file, enclosed with commenting marks. If a suitable realm is already defined, do the following:

   a. Identify the commented-out realm element that applies to the type of directory or database you plan to use, and then delete the commenting marks.

   b. Update the attributes in the new realm element with the details of your directory or database, using the instructions in Tomcat realms on the Apache site.

   If a suitable realm is not yet defined in the file, create a new realm element for your directory or database using the instructions in Tomcat realms on the Apache site.

8. Close and save server.xml.
9. Start the Tomcat service: CA Directory Webserver.

   Users in the directory or database can now log in to DXmanager.

How to Set Up Login Accounts in Tomcat

Note: CA recommends that you use this method for a test or demonstration system only. For a production system, use an existing directory or database for authentication.

To set up login accounts in Tomcat using the local XML file, follow this process:

1. (Optional) Enable hashed passwords in Tomcat (see page 58).
2. Create new login accounts for the users (see page 56).

Add a New Login Account in Tomcat

During installation, the administrative login account was created in tomcat-users.xml. You can add more accounts to this file, to allow other users to log in.

Note: CA recommends that you use this method for a test or demonstration system only. For a production system, use an existing directory or database for authentication.

Every login account in tomcat-users.xml must correspond to a user in the warehouse. The User ID must match the user name that you specify in the XML file.

Note: Before you begin, stop the Tomcat service: CA Directory Webserver. This disconnects any currently open sessions.

To add a new login account in Tomcat

1. Find the following file, and open it in a text editor:

   ```
   DXHOME/../dxwebserver/conf/tomcat-users.xml
   ```

2. Find the <tomcat-users> section, then add the following line:

   ```
   <user username="" password="" roles=""/>
   ```

3. Update the following values in the new line:

   username

   Specifies the name that the user enters when they log in to DXmanager. Make sure that this value corresponds to the User ID of a user in the warehouse.

   password

   Specifies the password for this user. If you have enabled hashed passwords, include the hashed value here.

   roles

   Specifies the roles for this user. This is a comma-separated list, so you can specify more than one role.

4. Save and close tomcat-users.xml.
5. Start the Tomcat service: CA Directory Webserver.

   A user can now use this account to log in to DXmanager. The account's roles define what tasks they can do.

Example: Assign Two Roles to a User

The following snippet of tomcat-users.xml shows the details for two users:

```
<tomcat-users>
<user name="Jenny Watts"
password="5460cdf0d7c4370224eb2be031b7c7eb724fa583"
roles="dxmanagerMonitor" />
<user name="Terry Morrow"
password="5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8"
roles="dxmanagerSuperUser" />
</tomcat-users>
```

In this example, Jenny Watts has the Monitor role, and Terry Morrow has the Superuser role. This means that Jenny can only monitor DSAs and customize monitoring functionality. Terry can perform all DXmanager tasks as he has Superuser access.

Note: The passwords in this example are hashed, which means that hashed passwords must be enabled in Tomcat.

More information:

Methods for Setting Up Login Accounts (see page 53)
Authentication Using Existing Accounts (see page 54)
Configure Authentication to a Directory or Database (see page 55)
Hashed Passwords for Login Accounts (see page 58)
Enable Hashed Passwords in Tomcat (see page 58)

Hashed Passwords for Login Accounts

By default, the Tomcat web server deals with clear-text passwords, regardless of the method you use for setting up login accounts.

If Tomcat uses an existing directory or database of accounts, then by default Tomcat expects to receive clear-text passwords. If your existing account repository already stores passwords as hashes, you should set Tomcat to accept hashed passwords.

If Tomcat uses a local XML file containing account details, then by default this file contains clear-text passwords, which can be read by anyone with access to the file. If you store the login accounts in the XML file, we strongly recommend that you configure Tomcat to use hashed passwords instead. This means that the passwords of any login account details that are stored in the local Tomcat file can be stored as a hash rather than in clear text.

To use hashed passwords in either the local XML file or in an existing database or directory, you must configure Tomcat to accept hashed passwords.

Enable Hashed Passwords in Tomcat

After you enable hashed passwords in Tomcat, you can include hashed passwords in either the local XML file of login account details, or in your existing directory or database.

Note: Before you begin, stop the Tomcat service: CA Directory Webserver. This disconnects any currently open sessions.

To enable hashed passwords in Tomcat

1. Find the following file, and open it in a text editor:

   ```
   DXHOME/../dxwebserver/conf/server.xml
   ```

2. Find the <realm> section, then add the digest and digestEncoding elements. For each of these, choose from SHA, MD2, and MD5. For example:

   ```
   <Realm className="org.apache.catalina.realm.UserDatabaseRealm"
   resourceName="UserDatabase"
   digest="SHA"
   digestEncoding=""/>
   ```

   Note: When you do not specify the digestEncoding element, Tomcat uses the platform default. If your directory or database uses a different encoding to the platform default, see the information about Tomcat realms on the Apache site for instructions.

3. Start the Tomcat service: CA Directory Webserver.

   You can now include hashed passwords in the login accounts for DXmanager.

DXManager Roles

A role defines what screens a user can see in DXmanager, which limits the tasks they can do. Every DXmanager login account has one or more roles.

You can assign each user to one or more of the following roles in ascending order of access:

dxmanagerMonitor

Users with the *Monitor* role can only monitor DSAs and customize monitoring functionality.

dxmanagerControl

Users with the *Control* role can start, stop, and initialize DSAs.

dxmanagerChange

Users with the *Configure* role can change the configuration of the backbone, and deploy the new configuration to the hosts.

dxmanagerSuperUser

Users with the *Superuser* role can set system properties. When you installed CA Directory, you created a password for a DXmanager user with the Superuser role.

Roles are hierarchical, so if an account has the superuser role, then it also has the abilities of all lesser roles.

Note: Users with more than one role assigned to them are granted the privileges of the role with the highest level of access.

The following table summarizes the privileges associated with these roles:

Privilege	Monitor	Control	Change	Superuser
Monitor DSAs	x	x	x	x
Set preferences	x	x	x	x
Start, stop, and initialize DSAs		x	x	x
Edit and save configuration			x	x
Deploy configuration to hosts			x	x
Set system properties				x

Troubleshooting DXmanager

If DXmanager cannot connect to the DXadmind on a host, then no information can be returned. This causes the following problems:

- You cannot deploy a new configuration to that host.
- You cannot view the status of DSAs on that host.
- You cannot stop, start, or reinitialize DSAs on that host.
- The maps show the gray Unknown icon ?.

To work out the cause of the problem, look for messages in the Alerts tab.

Cannot Create a DSA

Symptom:

I cannot create a DSA from the Namespace Map.

Solution:

If you cannot create a DSA from the Namespace Map, you may need to clear the Java Applet Cache.

To clear the Java Applet Cache:

1. Close the DXmanager browser window.
2. Select Start, Settings, Control Panel.
3. In the Control Panel, double-click on Java.
4. On the General tab, click the Settings button under Temporary Internet Files.
5. Click on the Delete Files button.
6. Ensure the Applications and Applets checkbox is selected.
7. Click OK.

The cache is now cleared and the DXmanager browser can be opened again.

DXmanager is Showing Unexpected Behavior

Symptom:

The DXmanager is showing unexpected or inconsistent behavior.

Solution:

If the DXmanager is behaving unexpectedly, you may need to log out and log back in to DXmanager.

Reset the DXadmind Configuration

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.

If there is a problem with the DXadmind configuration, you can use the *dxadmind setup* command to reset it.

In particular, DXadmind must be configured to trust the computer on which DXmanager is installed.

To reset the DXadmind configuration

1. Log in to the directory host.
2. Stop DXadmind, using the following command:

   ```
   dxadmind stop
   ```

3. Delete the file DXHOME/config/dsaconfig.xml, if it exists. This ensures that DXadmind is unaware of any existing DSAs.
4. Delete the initialization files of any DSAs that were previously configured on this host. These files are in DXHOME/config/servers.
5. In a command prompt, enter the following command:

   ```
   dxadmind setup trustedhost port [password]
   ```

   trustedhost

   Specifies the trusted management server. This is a computer that runs DXmanager.

   Note: You can specify the *trustedhost* using an IP address or using a host name. If you use the name format, then you must have reverse (ip-address to hostname) lookups enabled on the DNS.

   port

   Specifies the port that DXmanager uses to access DXadmind. This must be the same as the DXadmind port specified in DXmanager.

   password

   (Optional) Specifies the password that DXmanager uses to access DXadmind. This must be the same as the DXadmind password specified in DXmanager. You can either specify the password directly via the command line, or you can answer the password prompt.

6. Start DXadmind, using the following command:

   ```
   dxadmind start
   ```

Check Whether DXadmind Is Running

If DXadmind on a host is not running, DXmanager has no information about the DSAs on that host.

To check whether DXadmind is running

1. Log in to the host.
2. Open a command prompt and enter the following command:

   ```
   dxadmind status
   ```

   If DXadmind is running, the following response appears:

   ```
   dxadmind is running
   ```

3. If DXadmind is stopped, start it with the *dxadmind start* command.

Note: On Windows, the DXadmind service is named *CA Directory Administration Daemon - dxadmind.* On UNIX, the process is named *dxadmind start*.

Check Whether DXadmind Trusts DXmanager

If you are having problems with configuring or managing your directory backbone, you may have a problem with DXadmind on one or more of the directory hosts.

To check whether DXadmind trusts DXmanager

1. Log in to the host.
2. In a command prompt, enter the following command:

   ```
   dxadmind help
   ```

   The output lists the configuration of DXadmind on this host.

   In the output, look for the DXmanager Host. This is the IP address of the Management Server that DXadmind trusts.

3. Check that this really is the address of the Management Server.

Check Whether a Firewall Prevents Access to a Host

DXmanager may be unable to connect to DXadmind because a firewall prevents access to the DXadmind TCP port.

If this is the case, DXmanager reports the host as *unreachable*.

To fix this problem, configure the firewall to allow bidirectional access for the DXadmind port. (By default this port is 2123.)

Check That DXmanager Knows the Host's Network Address

If DXmanager does not know the correct network address of the Dxadmind host, it cannot connect to DXadmind.

If this is the case, DXmanager reports the host as *unreachable*.

Possible causes are as follows:

- There was a typing error when you set up the backbone configuration in DXmanager.
- The host is unavailable.
- DXadmind cannot establish the host name of the incoming connection.

Check that the host's address is correct and that DXadmind is running on the host.

If DXadmind is running and the DXadmind password and host details are correct in the DXmanager deployed configuration, then the problem is probably that DXadmind cannot establish the host name of the incoming connection. If you specified the DXmanager address to DXadmind as a host name (rather than as an IP address), then you need to have reverse DNS enabled. To solve this problem, enable the DNS to have reverse DNS (reverse lookup).

Change DXwebserver's Port Numbers

Check the logs in DXWEBHOME/logs for port conflicts. If there is a port number conflict, change the ports used by DXwebserver. To do this, modify the DXwebserver configuration file *server.xml*.

To change DXwebserver's port numbers

1. Find the *server.xml* file in *DXWEBHOME/conf*.
2. Open *server.xml* in a text editor.
3. Change one or more of the following port numbers:
    a. Change the standard port 8080, on which DXwebserver listens for non-SSL connections.

    ```
    <Connector port="8080" maxHttpHeaderSize="8192" maxThreads="150"
    minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
    redirectPort="8443" acceptCount="100" connectionTimeout="20000"
    disableUploadTimeout="true"/>
    ```

    b. Change the secure port 8443, on which DXwebserver listens for SSL connections. This is also used for the Coyote/JK2 1.3 connector port.

    You need to change the value 8443 in two locations in *server.xml*, as follows:

    ```
    <Connector port="8080" maxHttpHeaderSize="8192" maxThreads="150"
    minSpareThreads="25" maxSpareThreads="75" enableLookups="false"
    redirectPort="8443" acceptCount="100" connectionTimeout="20000"
    disableUploadTimeout="true"/>
    ```

    ```
    <Connector className="org.apache.coyote.tomcat4.CoyoteConnector"
    port="8443" minProcessors="5" maxProcessors="75" acceptCount="10" debug="1"
    scheme="https" secure="true" keystoreFile="__DXWEBSERVER_CERT_PATH__">
    ```

    Note: If you change the port number, the DSML sample server will not work.

    c. Change the shutdown port 8005, on which DXwebserver listens for the shutdown command, in the following section of *server.xml*:

    ```
    <Server port="8005" shutdown="SHUTDOWN">
    ```

4. Restart DXwebserver as follows:
    - On UNIX, use the following commands:

    ```
    dxwebserver stop
    dxwebserver start
    ```

    - On Windows, use the Control Panel to open the Services window, and restart the *CA Directory - DXwebserver* service.

DSA Console

The *DSA console* lets you connect to a DSA to give DXserver commands, receive trace information, and act as a user agent.

However, when the DSA shuts down, the configuration changes you made through the console are not saved.

We recommend that you define a console port for each DSA. Because the DSA can only have one console connection open at a time, this can prevent attacks from remote telnet sessions. This scheme uses the local host security. An administrator must have an account on the local computer to communicate with the DSA through this port.

More information:

Control DSAs by Using the DSA Console (see page 271)

How the DSA Console Works

When a DSA has its console port set, it accepts connections from *localhost*.

You need to specify the console port of a DSA to allow the DSA to accept local connections.

If you want to restrict access to the DSA through the console, you can also specify a console password. You can use SSL to encrypt communications on the remote connection.

When the DSA console is configured in *remote* mode, DSAs can accept connections from remote computers. You can use the console remotely if you have set up a remote console port and a password.

You can also use the remote console port to connect from *localhost*.

A DSA can accept only one console connection at a time, even if you have set up ports for both local and remote connections. This means you can prevent other console connections to a DSA by keeping a console connection open.

How the DSA Console Can Authenticate Using Directory Entries

You can set up the DSA to let users with an entry in the directory connect to the DSA console.

If you set this up, the following steps describe what happens when a user connects to the directory:

1. The user tries to connect to the directory, using the correct console port.
2. The user is prompted for the following information:
   - The user's DN in LDAP format (for example, *cn=Craig LINK,ou=Administration,ou=Corporate,o=democorp,c=AU*)
   - The user's password
3. The DSA checks that the DN and password match an entry.
4. The DSA checks whether the user or their role is listed in the *set dxconsole-users* command.

   If the user is listed, the user is now connected to the DSA console.

Note: We recommend using this feature through TLSClient to protect clear-text passwords.

More information:

Specify Which Users Can Connect to a DSA Console (see page 69)
Secure the DSA Console with TLSclient (see page 71)

Connect to a Local DSA Console

You can connect to a DSA locally if a console port has been set for that DSA.

To connect to a local DSA Console

1. Open a command prompt on the host on which the DSA is running.
2. Enter the following command:

   ```
   telnet localhost local-port-number
   ```

   local-port-number

   Specifies the console port number of the DSA to which you want to connect.

More information:

How the DSA Console Works (see page 65)
Configure a DSA Console Using DXmanager (see page 68)

Connect to a Remote DSA Console

You can connect to a DSA from any UNIX or Windows if a remote console port and password have been set for that DSA.

To connect to a remote DSA Console

1. Open a command prompt on a remote computer.
2. Enter the following command:

   ```
   telnet host-name remote-port-number
   ```

   host-name

   Specifies the name of the host on which the DSA is running.

   remote-port-number

   Specifies the remote console port number of the remote DSA to which you want to connect.

Important! If you do not want the password to be displayed when it is entered, turn local echo OFF on the Telnet session.

Example: Connect to a DSA Remotely

The following command connects to the DSA that uses the remote console port 19392 on the host *eagle*:

```
telnet eagle 19392
```

More information:

How the DSA Console Works (see page 65)
Configure a DSA Console Using DXmanager (see page 68)

Configure a DSA Console Using DXmanager

To connect to a DSA using telnet, you need to define a console port. You can also specify a password, and you can force all console connections to use SSL.

DXmanager lets you set the DSA console parameters for each DSA, and for all DSAs that serve a namespace partition.

To configure a DSA Console using DXmanager

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab, and choose the place you want to set up the console.
   - To set up a console for a particular DSA, display the Topology map, and navigate to the DSA.
   - To set up a console for all DSAs that serve a namespace partition, display the Namespace map.
4. Right-click the item for which you want to set up a console, and select Edit properties.
5. Find the Console settings (in the Console area), and enter the values as follows:

   Console port

   Specifies the port on which the DSA accepts connections from the local computer only. If this is not specified, there is no console for the DSA.

   Console password

   Specifies the password required for connection to this DSA. You must specify a password if you want to connect to this DSA from a remote computer. This password is optional for local connections.

   Remote console port

   Specifies the port on which the DSA accepts connections from remote computers.

   Remote console SSL

   Specifies that any console connections to the DSA from a remote computer are encrypted.
6. Click OK or Finish.
7. Save the changed configuration.
8. (Optional) Click Configuration, Deploy to update the hosts with the new configuration.

Specify Which Users Can Connect to a DSA Console

You can specify which users can connect to a DSA console.

To specify which users can connect to a DSA Console

1. Ensure that the DSA has a local or remote console port set up and working correctly.
2. Stop the DSA.
3. (Optional) Create one or more roles, containing users you want to give access to.
4. Add the following command to the DSA's *settings* configuration file:

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

   users

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

   roles

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

5. Save the changed configuration file.
6. Start the DSA.

Example: Set Up the Democorp DSA to Allow Directory Users to Connect through a console

This example shows how to allow users in the Democorp directory to connect to the DSA.

1. Add the following commands to the Democorp DSA's settings file:

   ```
   set dxconsole-users = <c AU><o Democorp><ou Corporate> <ou Administration><cn
   "Craig Link">,<c AU><o Democorp><ou Roles>;
   set role-subtree = <c AU><o Democorp><cn Roles>;
   set use-roles=true;
   ```

2. Use JXweb to add passwords to the following entries in the Democorp DSA:
   - cn=Nadia Kite,ou=Administration,ou=Corporate,o=Democorp,c=AU
   - cn=Craig Link,ou=Administration,ou=Corporate,o=Democorp,c=AU

3. Use JXweb to create a new role with one member as follows:
   a. Create the following entry with an object class of *groupOfNames*:
      - cn=Roles,o=Democorp,c=AU
   b. Add the following DN to the *member* field:
      - cn=Nadia Kite,ou=Administration,ou=Corporate,o=Democorp,c=AU
4. Reinitialize the Democorp DSA by running the following command as user *dsa*:

   ```
   dxserver init democorp
   ```

   You are now ready to test the DSA console login:
5. Try to use the DSA console to connect to the Democorp DSA using the following credentials:
   - User: cn=Craig Link,ou=Administration,ou=Corporate,o=Democorp,c=AU
   - Password: Use the password you created for Craig Link in Step 2.

   The login should work.
6. Try to use the DSA console to connect to the Democorp DSA using the following credentials:
   - User: cn=Nadia Kite,ou=Administration,ou=Corporate,o=Democorp,c=AU
   - Password: Use the password you created for Craig Link in Step 2.

   The login should work.
7. Try to use the DSA console to connect to the Democorp DSA using the following credentials:
   - User: cn=John Smith
   - Password: *password*

   The login should not work.

More information:

How the DSA Console Can Authenticate Using Directory Entries (see page 66)
Groups and Roles (see page 207)

Secure the DSA Console with TLSclient

The TLSclient utility establishes an encrypted tunnel between the remote console client and the DSA. This keeps the session secure. We recommend that you use TLSclient.

To set up TLSclient to work with the DSA Console

1. Stop the DSA.
2. Create the TLSclient configuration file and include the following line in the file:

   ```
   inPort outPort remoteAddress
   ```

   In this line, the variables stand for the following:

   inPort

   The port on the client computer that is used for tunneling.

   outPort

   The remote-console-port on the server.

   remoteAddress

   The host name of the server running the console you want to connect to.

   Here is an example for the sample Democorp DSA running on the server computer:

   ```
   19390 19395 hostname.ca.com
   ```
3. Save the new file on the client computer in this location: DXHOME/config/tlsclient/tlsclient.cfg.
4. Configure a DSA Console Using DXmanager (see page 68).
5. Install and start the TLS client:
   a. Install TLSclient to the system services using the following command:

      ```
      tlsclient install tlsclient-server-name -ca certificate-file
      ```
   b. Start the TLSclient instance using the following command:

      ```
      tlsclient start tlsclient-server-name
      ```
6. Start the DSA.
7. Test the connection:
   a. Open a Telnet window on the client computer.
   b. Connect to the inPort (defined in tlsclient.cfg) on the local computer.
   c. In the Democorp sample, this is 19390.
   d. Enter the console password when prompted.

      You now have a fully-functioning SSL-encrypted connection to the console on the server computer.

8. Verify that the connection works correctly:
    a. Start TLSclient with the debug option.
    b. On the DSA set tracing on using the following command:
    ```
    trace full;
    ```
    c. Use a packet-snooping application.

JXweb

JXweb is a general purpose LDAP-compliant directory browser and editor, which lets you access a directory through the Web.

JXweb lets you:

- Connect remotely to any directory that supports LDAP
- Browse the directory
- Update directory entries
- Change the directory tree structure
- View schemas
- View certificates
- Import and export bulk data

For more information, see the online help for JXweb.

Start JXweb

You can use JXweb to access a directory. This section describes how to start JXweb. After you start JXweb, you need to connect to a directory.

To start JXweb

1. Open a web browser.
2. Go to the following URL:

   `http://`*`hostname`*`:`*`portnumber`*`/jxweb`

   hostname

   Specifies the name of the computer on which JXweb is installed.

   portnumber

   Specifies the port number of CA Directory Web Server on which JXweb runs.

   Default: 8080

   The JXweb Connect page opens in your browser. You can now connect to a directory using LDAP or DSML.

Note: On Windows you can start JXweb by clicking Start, Programs, CA, Directory, JXweb.

Connect to a DSA Using JXweb

JXweb lets you connect remotely to any directory that supports LDAP.

After you connect to a directory you can view and update the directory data and structure.

To connect to a directory using LDAP

1. Do one of the following:
   - To connect to a directory for the first time in this session, start JXweb (see page 73). The Connect page appears.
   - To connect to a directory while you are browsing another directory, click Disconnect from the menu bar. Your current connection is disconnected, and the Connect page appears.
2. Enter the following data in the Connect page:

   Host

   Specifies the name of the computer on which the DSA resides.

   Port

   Specifies the port number of the DSA.

   Base DN

   (Optional) Specifies the base distinguished name of the DSA to which you want to connect (for example, *o=DEMOCORP,c=au*).

   User DN

   (Optional) Specifies the DN of your user account. If you use this, you must also specify the password.

   Password

   (Optional) Specifies the password of your user account. If you use this, you must also specify the user name.

   Use SSL

   (Optional) Specifies an SSL connection between the directory and the web server.

   Remember my connection settings

   Populates the Connect page with your last settings when you next open it.
3. Click Connect.

   JXweb is now connected to the directory.

DXtools

The DXtools simplify the management of directory data.

Together, these tools let you easily load and dump data for demonstration and prototyping.

Each tool can be run on a command line or in a script file.

Logs

The logs contain the output from a DSA.

You can use log files for the following purposes:

- Monitoring the health and performance of the directory
- Diagnosing problems
- Postprocessing to obtain statistics, billing information, and auditing information

Note: We recommend that you use only the logs that are actually required. Because a DSA has to write to the log files, keeping too many log files open can reduce performance.

More information:

Monitoring with Log Files (see page 256)

Traces

Tracing provides information that can help you to analyze the operations performed by the directory.

Tracing does not control what is logged to the summary, query, or stats logs.

In the DSA, the trace command controls the level of tracing that is sent to the DSA console, and to the trace log file, if the trace log file is open.

More information:

Monitoring with Traces (see page 261)

Alarms

Alarms are reports of critical events that should be monitored. Some triggers of alarms are:

- **A configuration problem**—For example, a DSA fails to start because it has the same listen address as one already running.
- **A usage problem**—For example, sending a DAP request to an LDAP port.
- **A system problem**—For example, running out of memory or disk space.

Unlike trace and log messages, alarm messages cannot be suppressed:

- Alarm messages are always written to the alarm log (which cannot be closed).
- When a console is open, alarm messages are sent to it.
- When an SNMP log is open, an SNMP trap is raised for every alarm.

More information:

Alerts Tab (see page 256)

Chapter 4: Set Up Schemas

This section contains the following topics:

Schemas

A *schema* is a formal definition of the contents and structure of the directory data. It governs where each entry can be placed within the directory structure, how entries are to be named, and what attributes each entry can contain.

When a DSA starts up, it reads the schema configuration files, which contain schema rules. The DSA enforces these rules on the directory data.

These rules define the following:

- The names of the various types of attributes that can exist in an entry
- The syntax (or data type) of each of these attributes
- Which attributes in each object class (a defined group of attributes) are mandatory and which are optional
- How each directory entry is named (for example, a person is named by his or her common-name attribute)
- Where the directory entry can be created in the namespace (DIT). For example, an *OrganizationalUnit* entry can only exist under an *Organization* entry.

The schema of a running directory is available to LDAP clients through the LDAP Version 3 mechanism of schema publishing. For more information, see Schema Publishing (see page 353).

Defining Schemas

You can configure all aspects of the schema, down to the level of attribute syntaxes compiled in the entry.

You should create the schema within configuration files (that is, in the DXHOME/config/schema directory) rather than dynamically using the console interface, as the latter only remains in effect while that DSA is running.

Registering an OID

Each object class or attribute type must be assigned a unique name and Object Identifier (OID). Generally, one OID is sufficient to meet all of your schema needs by adding another level of hierarchy to create new branches, or OID *arcs*. An OID arc is an OID that has been reserved for use as a container for defining additional OIDs.

You can obtain an OID from the IANA home page at http://iana.org. Other organizations that issue OIDs are ANSI (for US organizations), and BSI (for UK organizations).

Note: IANA's term for OIDs is *enterprise numbers*, as they are used primarily for Simple Network Management Protocol (SNMP).

Schema Prefixes

All attribute, attribute set, object class, and name binding definitions have a unique object identifier (for example, 2.5.4.6) that uniquely identifies that object.

When defining a schema, you can find many definitions on the same branch of the schema definition tree. You can define a schema prefix that replaces the branch of the tree in all subsequent definitions.

Example: Schema Prefix Definition

```
set oid-prefix x500attr  = (2.5.4);
set oid-prefix x500oc    = (2.5.6);
set oid-prefix x500aset  = (2.5.7);
set oid-prefix x500nbind = (2.5.15);

set attribute x500attr:3 = {
        name = commonName
        ldap-names = cn
        syntax = caseIgnoreString };
```

Given the previous definition, the prefix x500attr can replace any occurrence of the 2.5.4 portion of an object identifier. Thus, an attribute with the object identifier of 2.5.4.6 can also be represented as x500attr:6.

Without schema prefixes, the previous definition would read as follows:

```
set attribute (2.5.4.3)  = {
        name = commonName
        ldap-names = cn
        syntax = caseIgnoreString ;
```

Schema prefixes improve readability and reduce the chance of errors in the definition, especially when the object identifiers are long. For example, the object identifiers of each of the Quipu attributes are of the form 0.9.2342.19200300.99.1.x, making a prefix desirable.

Schema Groups

You usually define schema in a single definition file. All schema definition files reside in the schema configuration directory. When building your directory schema, you typically need definitions from several schema definition files. You can group these schema definition files together in a single file using the source command.

Example: Schema Configuration File

The following is an example schema.dxg file:

```
# Schema definition file - schema.dxg
source “x500.dxc”;
source “cosine.dxc”;
source “mhs.dxc”;
source “quipu.dxc”;
```

The initialization file (for instance, Democorp-Host2-democorp.dxi) sources this schema definition.

```
# DSA initialization file - Democorp-Host2-democorp.dxi
...
# SCHEMA
source “../schema/schema.dxg”;
...
```

Display a Schema

To display a schema, use one of the following methods:

- Use the *get schema* command

  For more information, see get schema Command.
- Use a directory browser such as JXweb.

Commands for Working with Schema

Use the following commands for working with schemas:

- clear schema command
- get schema command
- set attr-set command
- set attribute command
- set name-binding command
- set object-class command
- set oid-prefix command
- set syntax-alias command
- set unique-attrs command

For more information about these commands, see DXserver commands.

Changing the Schema for Attributes in Use

If an attribute, object class, or name binding has never been used in the directory, then you can change or remove the schema definition by modifying the schema configuration file.

If an attribute has ever been used within the directory, you can only change its schema definition using these instructions.

You cannot delete all the entries that have the attributes to be changed, change schema, and load back the data. Instead, you must reload the datastore with DXloaddb to remove the old schema definitions from the datastore.

Important! A single schema file may be used by many different DSAs. If you change a schema file, check that it works correctly with all of the DSAs that use that file.

Remember to regenerate the schema.txt file for the DAP tools after changing the DSA schema configuration.

Change the Schema Definition

Do the following to every datastore and DSA that uses the schema definition to be changed.

To change the schema definition

1. Stop all related DSAs.
2. Back up the system (data and configuration).
3. Use the DXdumpdb tool to dump the datastore to an LDIF file, using the following command:

   ```
   dxdumpdb -f LDIFfile DSA
   ```
4. Change the schema definitions.
5. Load the datastore from the sorted LDIF file with DXloaddb.
6. Start all the DSAs.
7. Verify the changes.

Changing LDAP Attribute Names

If an attribute is in use, and you only want to change its name, then alter the *ldap-names* part of the schema definition.

Local Schema

The X.500 standards enable the definition of local attributes, attribute sets, object classes, and name bindings in the schema in much the same way as described in Name Bindings and Aliases (see page 92).

Before defining local schema, you should check the existing published schema to determine whether the required attribute, object class, and name binding definitions already exist.

When you need to define additional schema, create an object identifier arc (1.3.6.1.4.1.3327.1 in the following example) and add the new schema under this arc. Use the set commands described previously to define the schema, and then include the newly created configuration file (.dxc) in the schema group configuration file (.dxg), used by the DSA.

The following example describes a single object class that can contain three attributes. CA created the object class so that you could add the additional attributes to an organizationalPerson object.

All the names in the following schema definition have the prefix *ca*. The use of an object identifier prefix in the name helps simplify attribute references by replacing the common portion of a complicated object identifier with a simple character string and helps identify related attributes.

Example: Local Attribute, Attribute Set, Object Class, and Name Binding Definitions

```
set oid-prefix caAttr   = (1.3.6.1.4.1.3327.1.4);
set oid-prefix caOclass = (1.3.6.1.4.1.3327.1.6);
set oid-prefix caAset   = (1.3.6.1.4.1.3327.1.7);
set oid-prefix caNbind  = (1.3.6.1.4.1.3327.1.14);
set attribute caAttr:0  = {
        name = caNearestPrinter
        syntax = caseIgnoreString
        description = "Local Printer Attribute" };
set attribute caAttr:1 = {
        name = caMobilePhone
        syntax = caseIgnoreString
        description = "Mobile Phone Attribute" };
set attribute caAttr:3 = {
        name = caAlternateContact
        syntax = caseIgnoreString
        description = "Local Contact Attribute" };
set attr-set caAset:0 = {
        name = caAttributeSet
        caNearestPrinter,
        caMobilePhone,
        caAlternateContact };
```

```
set object-class caOclass:0 = {
        name = caOrgPerson
        subclass-of organizationalPerson
        kind = structural
        may-contain caAttributeSet
        description = "CA Organizational Person Object Class" };
set name-binding caNbind:0 = {
        name = caOrgPerson-org
        caOrgPerson allowable-parent organization
        named-by commonName };
```

Attributes

An attribute is the foundation of directory information. It consists of a type and one or more values.

For example, in one entry, the attribute *commonName* could have the values *Rick* and *Richard*.

You define an attribute in the configuration file with information about its object identifier (OID), name, LDAP names, syntax, whether it is single-valued, whether its value can be modified, and a description.

There is no limit to the number of attributes or rules that you can define for a DSA or on the size of the attributes you can store in the DSA.

Example: Attribute Definition

```
set attribute x500attr:3 = {
        name                    = commonName
        ldap-names      = cn
        syntax          = caseIgnoreString
        description     = "Common Name attribute"
};
```

Attribute Names

Every attribute has a name.

You can also define one or more LDAP names for each attribute. The LDAP name defaults to the attribute name, so typically you need to define only the LDAP name when it is different from the attribute name.

For example, when you define an attribute using the following command, you can use *organizationName* or the shorter LDAP name *o* in other commands that need to reference the attribute:

```
set attribute (2.5.4.10) = {
    name        = organizationName
    ldap-names  = o
    syntax              = caseIgnoreString
};
```

Single Valued Attributes

By default, an attribute of each entry can have many values. You can restrict an an attribute to have only one value within an entry. You do this as part of the syntax specification of the attribute.The attribute is then called a single valued attribute.

Note: The single valued attribute is a characteristic of an attribute within each entry. Do not confuse this with the concept of unique attribute value, which is a run time check that compares the attribute values between different entries, and is not part of the schema.

Attribute Syntaxes

Attribute syntaxes define the format of basic information types.

New Syntaxes

To support new attribute syntaxes on demand, CA Directory DSAs accepts unknown attribute syntaxes and stores them in the directory. Intelligent matching rules are applied so you can search for them.

Changes to Attribute Syntax

Each DSA stores in the directory the names of attributes (and of attribute syntaxes) that occur in directory entries.

Note: A DSA does not store the names of *all* possible attribute names and syntaxes.

If you try to change the syntax of an attribute when an instance of that attribute already exists, the DSA does not allow this, and produces the following message:

```
** ALARM **: datastore attribute attribute has different syntax than schema
```

This approach maintains the integrity of the directory data when you change the configuration of the DSA.

Attribute Checking

When you load attributes (by using the *set attribute* command), the DSA checks their syntax. If you try to add attributes with values that do not comply with the attribute syntax, the DSA returns an error containing the attribute and the associated problem. For example if the syntax specifies numericString but the attribute value contains non-numeric characters, the DSA returns an error.

In the search service, the system ignores invalid attributes after the base object is found.

The DSA permits attributes of any size, so you do not have to define any attribute bound limits.

How CA Directory Stores and Returns Attribute Values

The DSA always stores and returns attributes exactly as you entered them. For example, a name given as *JohN citiZen* matches *John Citizen* and *JOHN CITIZEN* but the value *JohN citiZen* is always returned.

Syntax Aliases for Unsupported Attribute Syntax

If you add a new schema object definition that uses an attribute syntax that is not supported by CA Directory, use the set syntax-alias command.

Attribute Sets

Attribute sets let you easily group attribute definitions under a label so you can use them later in object class and other definitions.

Define attribute sets in the configuration file using the *set attr-set* command. The attribute set is given an object identifier and a definition. The definition consists of a name and a list of attributes or attribute sets that are contained in the attribute set being defined.

Example: Attribute Set

```
set attr-set x500aset:3 = {
        name = organizationalAttributeSet
        description,
        localeAttributeSet,
        postalAttributeSet,
        telecommunicationAttributeSet,
        businessCategory,
        seeAlso,
        searchGuide,
        userPassword
};
```

Attribute sets are a convenient way of grouping large numbers of attributes together for use in object class definitions. Attribute sets can contain other previously defined attribute sets.

Object Classes and Object Class Identifiers

An object class is an attribute of an entry. It specifies which attributes the entry can have.

An Object Class Identifier is a unique value that identifies an object class.

You define an object class by specifying its object identifier, a name, an alternate name, a subclass, an object class kind, a list of must-contain and a list of may-contain attributes or attribute sets, and a description.

Example: Object Class Definition

```
set object-class 1.3.6.1.4.5 = {
    name = organizationalUnit
    subclass-of top
    kind = structural
    must-contain
        organizationalUnitName
    may-contain
        organizationalAttributeSet
    description = "X.500 Organizational Unit Object Class"
};
```

Object Class Kind

The X.500/LDAP standards define the following three kinds of object classes:

- Abstract classes (for example, the object class top)
- Structural classes (for example, the object class inetOrgPerson)
- Auxiliary classes

If you do not define the kind for an object class, if the object class inherits top, the DSA assumes it to be structural class; otherwise the DSA assumes it to be auxiliary.

Abstract Classes

The abstract object class determines the structure of an LDAP directory. The object class *top*, for example, is the root object class from which all structural object classes are derived. It contains one mandatory attribute, *objectClass*, and because all entries inherit its attributes, it ensures that an object class defines these entries.

An abstract object class cannot stand alone in an entry. The entry must also contain a structural object class.

Structural Classes

Most of the object classes in a directory are structural, because they define what an entry is. They also impose rules on the entries that are stored beneath them. For example, the object class *organization* (o) may require that all objects stored beneath it belong to the object class *organizationalUnit* (ou). Other examples of structural object classes are *groupOfNames*, *inetOrgPerson*, and *person*.

Auxiliary Classes

An entry belongs to only one structural object class. However, an entry can also belong to one or more auxiliary object classes. An auxiliary object class adds attributes to entries already defined by a structural object class. An auxiliary object class cannot stand on its own in an entry. The entry must contain a structural object class. Unlike structural object classes, auxiliary object classes place no restrictions on storing an entry.

Object Class Checking

When adding an entry, the DSA includes all the attributes from all superclasses of the new entry's object class. For example, when an entry is created with the object class inetOrgPerson, it includes attributes from the inherited object classes (organizationalPerson, Person, top).

An object class can inherit attributes from more than one parent object class.

Object Class Storage

Every entry has an attribute *object class*. This attribute usually has multiple values, such as *organization*, *locality*, and *top*.

Configuring How the OC Attribute Is Stored

You can configure how the object class attribute (single value or multiple values) is stored within the directory.

By default, the DSA adds the object classes to the entry exactly as specified in the LDAP or DAP add request (for example, the object class attribute may have multiple OIDs). However, directory performance improves slightly when the object class attribute contains only a single value (the OID of the refined OC). However, this should not be the main influence over the object class design of attributes.

Where entries contain a single inherited object class and explicitly list all its superiors (their OC OIDs), the superiors can be removed, leaving the lowest-level object class in the hierarchy as a single OID value. Similarly, when entries are returned, the object class attribute can be automatically modified by the DSA to contain all the superior object classes.

This OC refinement information can be returned as OC OIDs because the directory maintains the object class structure rules that have been configured in its internal schema management control system.

For example, if a directory contains entries corresponding to people, then each entry can explicitly contain the following object classes within the datastore:

- inetOrgPerson
- organizationalPerson
- Person
- top

Pruning and Replacing Object Classes

It is more space-efficient to configure the DSA to prune all inherited object classes except the lowest (*inetOrgPerson*) when creating the entry, and to replace all the superior object classes *(organizationalPerson*, *Person*, and *top*) when returning the entry as a result of a client search.

The following boolean configuration settings control the pruning and replacing of object classes:

set prune-oc-parents

Removes redundant superior object classes when new entries are created.

set return-oc-parents

Replaces the superior object classes to entries returned to the client.

set add-oc-parents

Causes parent object classes to be added when entries are added. For example, consider adding a Democorp entry with the following class:

- inetOrgPerson

This control adds all the parent classes. As a result the entry holds the following classes:

- top
- person
- organizationalPerson
- inetOrgPerson

When prune-oc-parents or return-oc-parents is set, searching entries using an object class filter (for example, *oc=Person*) does not return any entries, because the specified object class of Person is not present in the data; it is added only to search results that contain the inetOrgPerson object class.

Check Structural Object Classes

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 use the *set check-structural-oc* command to enforce this.

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.

Name Bindings

Name bindings define where entries appear in the namespace. CA Directory requires a separate name binding for each parent-child relationship in the directory.

You can use multiple attributes together to name an entry, in which case you separate the attributes by commas. You can also specify additional optional naming attributes, using the keyword *optional*.

Example: Name Binding Definitions

In this example, a new definition (arbitrarily named org-country) states that you can place an *organization* entry under a *country* entry 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.

```
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: Advanced Name Binding Definitions

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

Name Binding Checks

The DSA checks name binding rules whenever you add or rename an entry. To enforce both the parent-child relationships in the directory, and the naming attributes used by a particular entry, name bindings are necessary. The system reports any name binding errors as follows:

```
Update Error: Naming Violation.
```

When you enable warnings (set trace = warn,...;), the system sends a message describing the reason for the name binding failure to the console.

There is one exception regarding name binding checks. When an object is added under the naming context (or prefix) of a DSA, then no name bindings need exist. This facilitates the changing of the directory's naming context without the need to change associated name binding definitions. In this situation, CA Directory gives the following message:

```
Relaxed name bindings under root.
```

Name Bindings and Structural Object Classes

When an entry has more than one object class, the DSA looks through its list of name bindings to ensure that one exists between one of the structural object classes of the parent and one of the structural object classes of the entry containing the appropriate naming attribute. It then uses this structural object class to find a name binding and ignores all other auxiliary object classes.

The DSA permits modification of the structural object class only when a name binding satisfies the parent-object relationship and the object is a leaf entry.

Name Bindings and Aliases

The DSA lets you create an alias entry in place of any valid object if you name it using the same attribute that an equivalent name binding rule permits. Name bindings for aliases are automatic, and you do not configure them manually.

You do not need to define an object-alias name binding for every part of the DIT where you can place an alias.

For example, when there is an org to orgUnit name binding, the DSA lets you place an alias under an organization object when you name it using an organizationalUnitName attribute.

More information:

Set Up Aliases (see page 393)

Considerations for Schemas That Do Not Support Name Binding

When a schema is imported from a directory that does not support name bindings or structure rules, CA Directory can operate without name bindings. To enable this functionality, add the following command to the settings file:

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

You can retrieve the state of this flag by using the *get user* command.

Chapter 5: Set Up Replication

This section contains the following topics:

What Is Replication?

In a replicated directory, information that is stored in one part of the directory is also stored elsewhere as one or more copies.

Replication involves copying data and ensuring that the copies are synchronized.

More information:

Regions and Replication (see page 47)

Benefits of Replication

Replication is deployed for one or both of the following reasons:

Improved availability

Replication can make a system more robust by enabling the directory on one server to failover to an alternative server. Applications can continue working, because as there are alternate DSAs or servers in the network that can serve the same parts of the namespace.

Improved performance

Replication can place frequently accessed information closer to where it is needed, which improves response times and allows the load to be shared between DSAs.

Distribution and Replication Are Different

In a distributed directory, each piece of data is stored on only one server. The namespace is partitioned and each partition is stored on a different server.

In a replicated directory, each piece of data is held on more than one server. The entire namespace is stored in more than one place.

You can use both replication and distribution in the same system. The namespace is partitioned over many different servers, and at least some sections of the namespace are also copied and stored in more than one place.

Types of Replication Supported on CA Directory

CA Directory supports the following methods of replication:

Multiwrite

Multiwrite replication is a mechanism for replicating updates to a number of DSAs to ensure that they are synchronized. When a DSA receives an update, it updates its own data and then sends the update to its peers. If a peer DSA cannot be reached, the updates are queued and replayed when the DSA becomes available.

DISP

*DISP (*Directory Information Shadowing Protocol) is defined in the 1993 X.525 standard. DISP lets you replicate information in OSI-conformant directories, which permits copying of directory information from one DSA to another using a standardized procedure and protocol.

Multiwrite replication with DISP recovery ('Multiwrite-DISP')

Multiwrite-DISP replication is a replication scheme that uses multiwrite replication for real-time updates and DISP for recovery.

Multiwrite and DISP have complementary strengths: multiwrite replays updates to another DSA, and is fast and secure for small differences. However, if multiwrite has more updates (to be replayed to another DSA) than it can fit in its queue, it stops using the queue and drops all update information for that DSA, so it cannot cope with a DSA being down indefinitely. By contrast, DISP compares the current states of two DSAs and can handle a DSA being down indefinitely.

Multiwrite-DISP is the recommended method of replication with CA Directory because it combines the efficiency of multiwrite when DSAs are online, with the robustness of DISP to allow DSAs to recover after being offline.

Multiwrite Replication with DISP Recovery (Multiwrite-DISP)

Multiwrite-DISP replication is a replication scheme that uses multiwrite replication for real-time updates and DISP for recovery.

It is the recommended method for replication.

How Recovery Works in Multiwrite-DISP

When multiwrite replication with DISP recovery is enabled, multiwrite queues are not maintained for offline DSAs; any DSA that goes offline will rely on DISP recovery instead.

To enable multiwrite replication with DISP recovery use the following:

```
set multi-write-disp-recovery = true;
```

Note: For the purposes of explanation, we call the DSA that is always available DSA 1, call the DSA that was unavailable and comes online, DSA 2.

When an unavailable DSA comes online, the following occurs:

1. The multiwrite queue on DSA 1 for DSA 2 is enabled.
2. DISP on DSA 1 performs the resynchronization by calculating and sending an update for the period in which DSA 2 was offline. While this is occurring, DSA 1 queues any further updates to DSA 2.
3. When the DISP update completes, the multiwrite queue on DSA 1 is replayed to DSA 2.
4. Multiwrite operation resumes.

Note: There are no explicit DISP agreements; all resynchronization and reconciliation is automatic.

Benefits of DISP Recovery in Multiwrite-DISP

Enabling DISP recovery generally avoids manual resynchronization of the datastores.

DISP recovery works with multiwrite replication by handling all recovery operations. In effect, fast multiwrite is used during uptime and DISP is used only for recovery after a downtime.

DISP recovery has the following advantages:

Stable Calculation of Required Updates

Updates are calculated from the original datastore, not queued in memory. This means that recovery can survive restarts of a master (although you can lose the in-memory queues).

State-based

Resynchronization uses state-based updates. When there are a large number of replays, this is more efficient than replay-based recovery.

Reconciliation

DISP processing supports automatic conflict resolution using a last-write-wins rule. Managing conflicts is often a problem for replay-based systems because the order of updates from different masters can be critical.

Note: DISP recovery uses the highest authentication level that is available. Only specify the use of SSL authentication level (*ssl-auth* in a knowledge file) if you have set up the necessary SSL certificates for the DSAs in the region.

Enable Multiwrite-DISP

To turn on multiwrite with DISP recovery, use DXmanager, or add the following command to the DSA settings file:

```
set multi-write-disp-recovery = TRUE;
```

Granularity of DISP Reconciliation

When DISP resynchronization starts, it propagates all the changes to the DSA since the last successful multiwrite update.

In a multimaster system, it is possible for some changes to clash. In practice this is exceedingly rare. The window for discrepancy is small and all running master DSAs keep themselves synchronized using multiwrite. However, CA Directory resolves such conflicts automatically, using a last-write-wins rule. The granularity of reconciliation is an X.500 object.

Example: Conflict Resolution

If DSA 1 updates an object's surname attribute at time T, and DSA 2 updates an object's *phoneNumber* attribute at time T+1, then the final object after resynchronization is the object contained in DSA 2. This does not have the changes to the surname.

Add a DSA to a Multiwrite-DISP System

You might want to add a DSA to a system, for example, to increase reliability.

To add a DSA to a multiwrite-DISP system

1. Create the new peer DSA, for example, by using the DXnewsdsa tool.
2. Configure the new DSA to enable multiwrite, for example, by following the instructions in Enable Multiwrite Replication Using Configuration Files (see page 106).
3. Configure the new DSA to enable DISP recovery, for example by adding the following command to the DSA configuration file:

   ```
   set multi-write-disp-recovery = TRUE;
   ```
4. Reset the times in the DISP files on each host for all peer DSAs (including the newly added one) using the following command:

   dxdisp *newDS*

   This command updates the .dp file to prevent all other peer DSAs trying to replicate all their data to the new one.

   Note: For more information on these commands, see the *Reference Guide*.
5. Take a consistent snapshot copy of one of the running DSAs.

   Note: You could use the DXdumpdb tool instead, but the procedure for using DXdumpdb is more involved than using this online dump method.
6. Rename and move the file that is produced from the online dump to form the datastore file for the new DSA, as follows:

   Rename the database file *existingDSA*.zdb to *newDSA*.db
7. Start the new DSA.

Transaction Log and Data Recovery

CA Directory keeps a transaction log of the updates it makes to a datastore. You should configure the transaction log based on your requirements for safety versus performance:

- **Disable transaction logging**—Provides the highest performance, but if the DSA exits abnormally, or there is a power failure, you will have to perform disaster recovery to restore the datastore from a backup.
- **Enable logging without flushing**—Provides a performance that almost parallels that of a disabled log. If the DSA exits abnormally, you can simply restart it and recovery is not required. However, if there is a power outage, recovery is necessary.
- **Enabled logging with flushing**—The safest setting as the DSA can be restarted after an abnormal termination or power failure and recovery is not required. However, there will be around 100 updates per second, instead of around 10,000 per second if flushing was disabled.

Note: Just setting the transaction log to flush to disk may not be enough for safety. This is because many operating systems, depending on configuration and disk systems, may report that the data has been written to disk some time before the transfer actually takes place. Refer to your OS documentation for more information about write behind settings.

Single Server Recovery

If you are running a single server (with no redundancy), enable the transaction log. For maximum security, enable flushing of the transaction file. If this impacts performance too greatly, and recovering from backups is an option, you can turn flushing off.

You need to perform disaster recovery for a single server if any of the following occurs:

- Loss of a server
- DSA exits abnormally and there is no transaction file
- A power failure or a critical error in the operating system and the transaction file is not being flushed
- A disk failure

To perform disaster recovery in these cases, recover from backups.

Peer DSAs Recovery

If there is redundancy in the directory configuration, it is possible to recover DSAs from their peers. If a DSA terminates abnormally, and there is a transaction file, the DSA may simply be restarted. If it had multiwrite operations queued, these will be lost. Therefore, the configuration of these DSAs should:

- Enable multiwrite-DISP
- Enable the transaction file

If a DSA terminates normally or abnormally, it may simply be restarted. The transaction file allows the DSA to make the datastore consistent and multiwrite-DISP ensures that the DSA is synchronized with its peers.

Note: For more information, see How Recovery Works in Multiwrite-DISP (see page 97).

You need to perform disaster recovery if any of the following occurs:

- There is no transaction file
- There was a power outage or an operating system failure
- There was a disk failure
- The DSA has been down for a long time (or it is a new DSA)

How to Perform a Disaster Recovery

If the conditions for a disaster recovery for your CA Directory implementation occurred, you can no longer rely on automatic recovery from a transaction file (single server) or from a peer DSA (multiwrite-DISP). Instead, you need to recover your data from a backup.

To recover your data in the case of a disaster, do one of the following.

- For a single server:
  1. Locate your last backup copy.
  2. Delete the transaction log (.tx file).
  3. Rename the backed up files to *server.type*

     To do this, remove the prefix z from the filename extension.
  4. Restart the DSA

- For multiple peers:

  Note: The steps below assume that you have multiple peers and peer B was unavailable for a long time. The procedure for adding a new peer is very similar.

  1. Run the following command on all hosts with peers, including host B:

     ```
     dxdisp B
     ```

     This sets the last update time to now (T).

  2. Remove the transaction log file on B, if it exists.
  3. Take a snapshot copy of any one of the running DSAs.

     The online dump creates a backup file: *.zdb

  4. Copy the backup file to B.
  5. Rename the copied file by removing the *z* file extension prefix.

     You should now have: *server*.db

  6. Start B

     The DSA peers send all changes since T to B to bring it in sync.

More information:

Add a DSA to a Multiwrite-DISP System (see page 100)
Online Datastore Dump (see page 279)

Multiwrite Replication

Multiwrite replication is a good choice for a system that includes applications requiring real-time replication.

To use multiwrite replication successfully, your network links, computers, and power supplies should be reliable.

Important! Treat system crashes and unreliable networks as disaster scenarios. Recovery requires manual reconciliation between datastores.

How Multiwrite Replication Works

Multiwrite replication uses the Directory System Protocol (DSP), or LDAP via Dxlink to an LDAP server, to send updates in real time to all replication peers in the same region. When a client makes an update request, that update is applied immediately to the local DSA, and then to all other DSAs. The client receives a confirmation response only after all DSAs have responded.

Example: A Simple Multiwrite System

The following diagram shows these steps:

1. The client sends an update request to the directory.

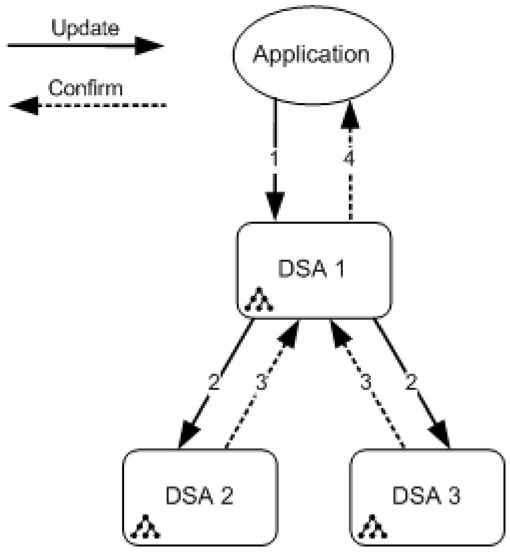
   DSA 1 receives the request and immediately applies the update to itself.

2. DSA 1 sends the update request to its peers, DSA 2 and DSA 3.

3. DSA 2 and DSA 3 each apply the update to itself, and then send an update response to DSA 1.

4. After receiving an update response from both peers, DSA 1 sends an update confirmation to the client.

   Any client can now query any DSA and it gets the same response, because the update has been made to all DSAs.



Multiwrite Queues

If a peer DSA is offline, the sending DSA puts new updates for the peer in a queue, and periodically tries to connect to the peer.

When the peer DSA comes back online, the queued requests are sent in the order that they were processed locally.

While the peer DSA remains offline, the queue grows. The DSA raises an alarm, and writes to the alarm log, at 60%, 70%, 80%, 90%, and 100% of queue capacity.

If the queue becomes full, the peer DSA ignores the unavailable DSA. It discards all the queued requests for the unavailable DSA and temporarily drops it from the multiwrite set. To bring this DSA back into service, you must resynchronize the DSA datastores manually, and restart the DSAs.

If a DSA has attempted a multiwrite operation, the *get dsp* console command returns one of the following states:

Failed

Indicates that the multiwrite DSA is not responding to an update. Updates for that DSA are held in the multiwrite queue until the DSA responds. The queue is retained until the multiwrite queue size is exceeded.

Recovering

Indicates that the DSA has become available and still has old updates in its queue.

OK

This is the normal multiwrite DSA status.

How to Set Up Multiwrite Replication

If you want to ensure that a region continues to service a namespace, even if one DSA fails, you need at least three DSAs in each region. This is because if one DSA fails, you may need to take a second DSA offline to resynchronize the failed DSA.

Enable Multiwrite Replication Using DXmanager

By default, DXmanager sets all DSAs that share the same namespace partition and are in the same region, to replicate each other with multiwrite. This means that using DXmanager makes it simple to enable multiwrite replication.

To enable multiwrite replication using DXmanager

1. Decide which DSAs to include in the multiwrite system.
2. Use DXmanager to create these DSAs as part of the same region.
3. Ensure the datastores of each DSA hold synchronized information.

Enable Multiwrite Replication Using Configuration Files

If you do not use DXmanager and you want to enable replication, you need to use configuration files.

To enable multiwrite replication using configuration files

1. Add the DSA flags multi-write and no-service-while-recovering to the shared knowledge, as follows:

   ```
   set dsa dsaname =
   {
   ...
   dsa-flags = multi-write, no-service-while-recovering ...
   ...
   };
   ```

2. (Optional) Define the multiwrite group in the DSA's shared knowledge:

   ```
   set dsa dsaname =
   {
   ...
   multi-write-group = group-name
   dsa-flags = ...
   ...
   };
   ```

Set the Retry Interval

If a DSA fails to bind with a multiwrite peer, it tries again. By default, the interval between tries is 60 seconds.

To set the retry interval, include the following command in the DSA knowledge:

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

Prevent Multiwrite Queues Being Lost When the DSA Stops

If you stop a DSA that has multiwrite queues, those queues are lost. This means that you could lose data. CA Directory helps prevent this loss by letting you specify the Wait for Multiwrite configuration item.

If this configuration item is set to *True* and you stop the DSA, the DSA stops processing but it does not exit while it has multiwrite queues that contain data.

To prevent multiwrite queues being lost when the DSA stops, using DXmanager

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Click the Maps tab, and choose the place where you want to set Wait for Multiwrite to *True*:
   - To set this value for a particular DSA or for the entire backbone, display the Topology map.
   - To set this value for all DSAs that serve a namespace partition, display the Namespace map.
3. Right-click on the item for which you want to set Wait for Multiwrite to True, and click Edit Properties.
4. Find the Replication area, and set Wait for Multiwrite to *True*.
5. Click OK or Finish.
6. Save and deploy.

More information:

How to Stop a DSA (see page 265)
Force a DSA to Stop Using a Command Line (see page 270)
Force a DSA to Stop Using the DSA Console (see page 272)

Change the Maximum Queue Size

The default maximum size of a multiwrite queue is 20,000 updates for each peer. You can gauge the required size by looking at the stats log and examining the number of updates and recovery time.

To change the maximum queue size, use the following command:

```
set multi-write-queue = queue-size;
```

queue-size

Defines the maximum size of the multiwrite queues.

Display the Multiwrite Status of a DSA

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

The output of this command displays the following, if they exist:

- Multiwrite queues
- Recovery notification list

To display the multiwrite status of a DSA, enter the following command on a DSA console:

```
get dsp;
```

Recovery in Multiwrite

A multiwrite update can fail. Usually, this is because a DSA is down or otherwise of offline. When the DSA comes back online again, the system must be recovered.

With multiwrite replication, there are two ways you can recover the system: multiwrite recovery and DISP recovery. If recovery is with multiwrite, the replication scheme is just called multiwrite. If recovery is with DISP, the replication scheme is called Multiwrite-DISP (see page 97).

Stop a Recovering DSA Sending Out-of-date Information to a Client

If a DSA has been offline, when it comes back online, its data is out of date.

To prevent a recovering DSA from sending out-of-date information to a client, use the *No Service While Recovering* setting on all the DSAs in a multiwrite system.

No service while recovering

Defines the behavior of the DSA while it is recovering from an outage. If this is selected, then a recovering DSA accepts updates from peers only, and does not accept other client operations. This prevents clients from accessing out-of-date data.

Multiwrite Replication with Multiwrite Recovery

Multiwrite is based on the idea that the multiwrite DSAs are usually functioning and connected. If one of the DSAs in the region shuts down or becomes disconnected, any updates are queued in another DSA's memory until the offline DSA becomes available once more.

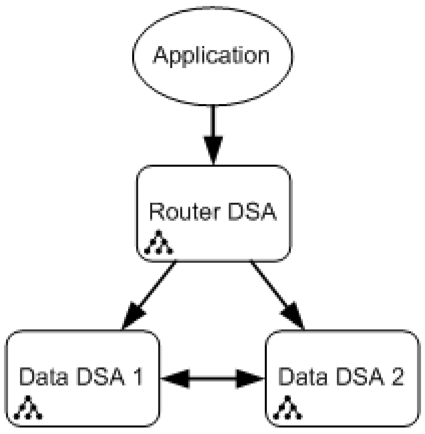
After the DSA puts the update request in a queue, it sends a confirmation to the client. In effect, multiwrite reverts to a write-behind scheme until the offline DSA becomes available.

Important! A queued update is stored in memory only, and is lost if the DSA holding the queue is restarted.

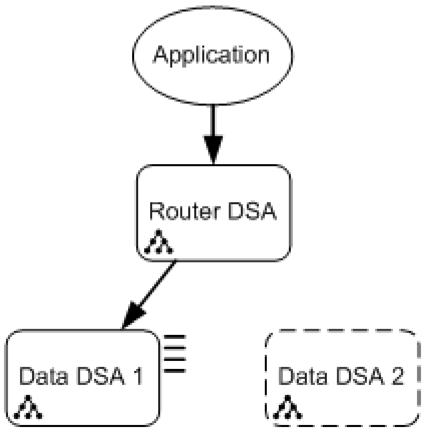
The following diagrams show how a DSA in simple multiwrite system recovers.

1. The system is functioning correctly.

   A single router DSA passes client requests to two data DSAs, which replicate all changes to each other.



2. Data DSA 2 goes offline.



While DSA 2 is down, the following happens when the client application makes an update request:

a. The router DSA passes the update request to DSA 1.

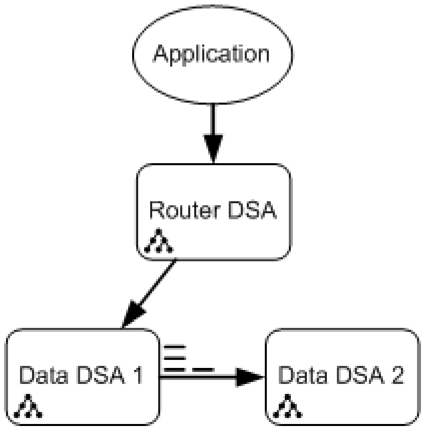
b. DSA 1 makes the update to itself and queues the update for DSA 2.

   DSA 2 is now out-of-date.

3. DSA 2 comes online again, as follows:

   a. DSA 2 starts up in recovery mode, which means that it can receive binds only from its peer, DSA 1.

   b. DSA 1 sends updates from its queue, in the order that it queued them, to DSA 2, as follows:



   c. When the queue is empty, DSA 1 sends a notification to DSA 2 that the data is synchronized. This switches DSA 2 out of recovery mode, returning it to service.

Manually Resynchronize a DSA in a Multiwrite System

Sometimes you might want to manually resynchronize a DSA, rather than rely on multiwrite recovery. For example, if a DSA has been offline for a while, its data might be so out of date that it is quicker to manually resynchronize it than to apply updates.

For the purposes of explanation, assume we have three DSAs:

- DSA 1. This DSA is up-to-date and keeps working as a directory server throughout the resynchronize process.
- DSA 2. This DSA was working fine and is up-to-date at the start of the synchronization process, but we will take it off-line to use it as the source for loading DSA 3.
- DSA 3. This one has been offline. We are going to load it with up-to-date information from DSA 2.

Note: When you are deciding which of the working DSAs to be the source for loading the offline DSA, choose the DSA that does not have a multiwrite update queue for the other DSA. This means you will not lose data when you take it offline.

To manually resynchronize a DSA in a multiwrite system

1. Stop DSA 3 if it is not already stopped.
2. Reinitialize DSA 1 using the init command.

   This causes DSA 1 to restart its queue for DSA 3. It does not affect its queue to DSA 2.
3. Stop DSA 2.

   When the DSA is stopped, its datastore is not updated while you extract the data from it.

   Important! Any multiwrite queues in this DSA are deleted, so check these queues before you stop it. You may need to use the command *forcestop*. If do this, you will lose data.
4. Copy the datastore files from DSA 2 to DSA 3.
5. Restart DSA 2 and DSA 3.

   Note: Be sure these DSAs start with the *No Service While Recovering* setting set to true. By default, DXmanager sets this setting to true.

More information:

How to Stop a DSA (see page 265)

Test Whether Two Datastores Are the Same

You may want to test whether two datastores are the same. For example, you might suspect that the synchronization has failed between two datastores.

To test whether two datastores are the same

1. For each datastore, shut down its DSA.

   The datastores do not change while you extract the data from them.

2. Use dxdumpdb to dump the datastores to LDIF files, as follows:

   ```
       dxdumpdb -f old.ldif old_dsa
       dxdumpdb -f latest.ldif latest_dsa
   ```

   Note: If you do not intend to resynchronize the datastores using these snapshots, then you can restart the DSAs now.

3. Use ldifsort to sort the LDIF files, as follows:

   ```
   ldifsort old.ldif old_sorted.ldif
   ldifsort latest.ldif latest_sorted.ldif
   ```

   You now have two LDIF files that can be compared.

4. Use ldifdelta to compare the two LDIF files, as follows:

   ```
   ldifdelta old_sorted.ldif latest_sorted.ldif
   ```

DISP Replication

CA Directory implements the 1993 Directory Information Shadowing Protocol (DISP), which lets you replicate information in OSI-conformant directories. This implementation supports the following:

- DISP routing
- Shared configuration
- On-demand and periodic updates

X.500 defines a master-slave replication scheme using DISP. In this scheme, any update that succeeds locally is forwarded to other DSAs according to any controlling bilateral agreement with those other DSAs.

Features of DISP Replication

DISP (Directory Information Shadowing Protocol) is the standard replication protocol for X.500. Some important features of DISP are:

- Incremental replication, to optimize network and resource usage.
- Scheduled and on demand updates.
- Total updates, which allow for setup and reset.

DISP replication may cause large updates which make the DSAs appear to be unresponsive.

To help make updates between DSAs in a Multi-Write DISP configuration as fast as possible, the system will assess the number of updates that need to be sent to a peer DSA and if that number is greater 5% of the total number of entries on that DSA, it will send updates in smaller packets of data.

Shared DISP Configuration

You can achieve replication between two DSAs using point-to-point DISP. To do this, share the DISP configuration between the DSAs.

When two DSAs share a configuration that includes a DISP agreement, the system performs automatic DISP updates as determined by that agreement.

Note: CA Directory supports the push DISP scenario, not the pull DISP scenario. In DISP replication systems, the master is always the supplier and the slave is always the consumer.

DISP Agreements

A DISP agreement is a set of statements that defines the DISP replication between two DSAs. When two DSAs share an agreement, one of the DSAs is the master and the other is the slave. A DSA can have a number of agreements relating to one or more remote DSAs.

Define each agreement with an agreement identifier and an agreement version.

View DISP Agreements

To view the details of *a* DISP agreement, use the following command:

```
get agreement id.version;
```

To view the details of *all* DISP agreements, use the following command:

```
get agreements;
```

Set a DSA to Use DISP Replication

If you want a DSA to take part in DISP replication, you must set it up to use DISP replication.

To set a DSA to use DISP replication

1. Create a DISP agreement, using the *set agreement* command.
2. In DXmanager, ensure that the DSA's configuration includes a DISP PSAP definition.

This DSA remains active, awaiting incoming DISP protocol requests.

Example: DISP Replication

To configure the replication in the Democorp example, perform the following steps:

1. Stop the DSAs.
2. Ensure that the datastores of each DSA contain the same data.
3. Create the DISP agreement in the configuration file (agreement.dxc) in the replication directory of both DSAs, using the following command:

   ```
   set agreement 1.1 =
   {
      initiator = Democorp
       responder = Consumer
       area      = <countryName AU><organizationName Democorp>
   };
   ```

4. Ensure that the configuration for the Slave (Consumer) DSA configuration includes a DISP PSAP definition.
5. Edit the Democorp master DSA initialization file (.dxi) to source the replication agreement, using the following command:

   ```
   # replication agreement
   source "../replication/agreement.dxc";
   ```

6. Edit the Democorp slave DSA initialization file (.dxi) to source the replication agreement (agreement.dxc), using the following command:

   ```
   # replication agreement
   source "../replication/agreement.dxc";
   ```

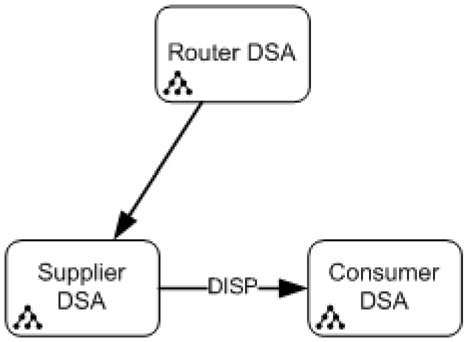
7. Start the DSAs, for example by using the following command:

   ```
   dxserver start all
   ```

Note: You can also consider setting the slave DSA to be a shadow.

DISP Routing

CA Directory supports the concept of a DISP routing that lets you route DISP through one or more intermediate DSAs. This is especially useful for firewall applications, as shown in the following diagram:



To include a DSA that does DISP routing in a DISP replication process, all three DSAs must have an agreement that identifies the DSA that does the DISP routing. All three DSAs can share the same DISP configuration.

Manually Perform a DISP Update of DISP

You can manually perform a DISP update when an agreement has been defined, by using the following command:

```
update agreement id.ver;
```

The agreement can define an incremental or a full update. An incremental update refreshes all entries that have changed since the last DISP update and since the DSA started.

Selective Shadowing

You can restrict the data that a shadow master replicates to a shadow slave. There are three ways to achieve this:

- Specify the replicated area
- Specify a subtree filter for the replication area
- Specify the set of attributes to be shadowed

Replicate an Entire Subtree

To replicate an entire subtree, use the *area* parameter in the DISP agreement to specify the top of the subtree.

The Filter Parameter in a DISP Agreement

You can use the optional *filter* parameter in the DISP agreement to use an X.500 filter to specify entries.

If you use the subtree filter, you should base the restrictions on object class, not on attributes. If the subtree filter contains anything other than comparisons of object class values, this can cause the shadow DSA to have incorrect data. This happens if the entry is modified on the master DSA and so no longer matches the search filter. In that case, subsequent DISP updates do not contain this entry, and leave the shadow DSA with unmodified and incorrect data.

Potentially the subtree filter could cause problems when shadowing to other vendors' directories, if for example, a full refresh is requested, or if an entry is shadowed but its parent entry does not exist on the shadow slave. CA Directory overcomes this potential problem by automatically creating Directory Specific Entries (DSE) that have a name but no attributes.

The Attributes Parameter in a DISP Agreement

Use the optional *attributes* parameter in the DISP agreement to specify the set of attributes to be shadowed.

Each element of *attrSelectionList* is an *attrSelection* element, which specifies the attributes that the shadow supplier is to select for shadowing. Specification of attributes for an object superclass also applies its subclasses. If the class is omitted, the selection applies to all classes.

When using the exclude specification, any attributes contained in an entry that are not explicitly excluded are implicitly included. Attributes are implicitly included where all-attributes is specified.

The attribute object class, all attributes that are described in the schema as *must contain*, and all operational attributes, are replicated unless they are listed in an exclude list.

Where entries belong to more than one of the specified classes, the specifications are cumulative. When there are conflicting specifications, include has priority over explicitly excluded attributes and exclude has priority over implicitly included attributes.

A selective DISP update can include entries that do not satisfy the schema requirements of the shadow consumer. For example, mandatory attributes need not be present in an entry contained in the DISP update.

With CA Directory, such an update is allowed because the master DSA verifies the schema of the complete entry and allows the partial replication to the shadow DSA. This may, however, cause problems when replicating to other vendors' directories.

If you flag the DSA as a shadow (so that it is read-only for non-DISP operations), the fact that not all attributes are present for a particular entry is of no consequence.

Chapter 6: Set Up Distribution and Routing

This section contains the following topics:

How Distribution Works

In a *distributed* directory, many DSAs cooperate to form a single namespace. Parts of the namespace are served by different DSAs. An application connected to any DSA can search the entire namespace.

From the client's point of view, the directory is a single entity, because every request is routed to the appropriate DSA within the namespace.

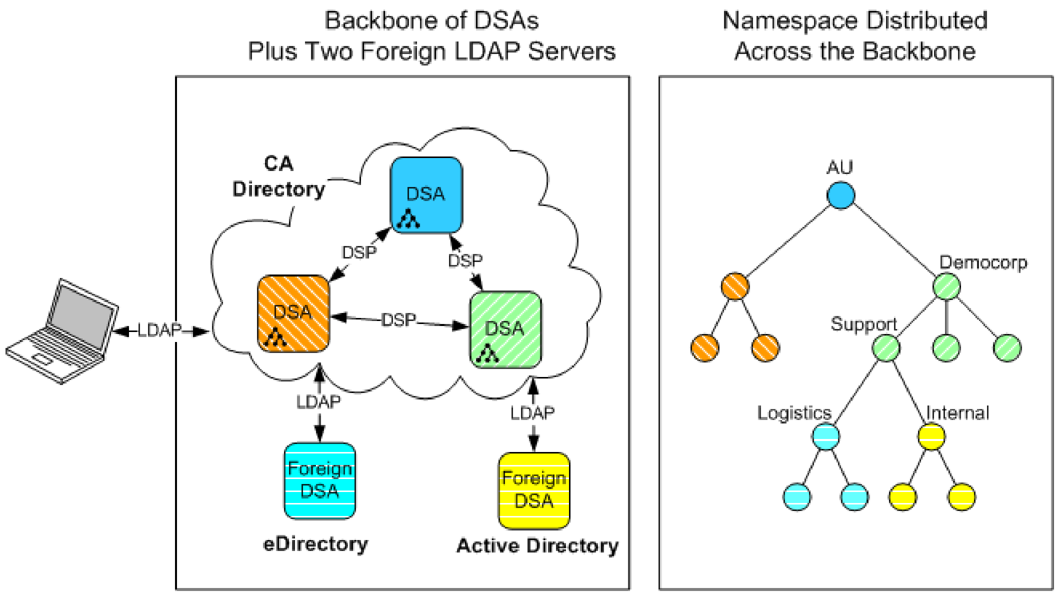
The DSAs may be all on the same computer, or they may be on different computers.

A distributed directory is useful to organizations with offices in several locations, because each office can maintain its own directory, and yet to a client it appears as just one directory. It is also useful for large directories because you improve performance when you share the data across different servers.

A distributed directory can incorporate LDAP servers. CA Directory fully complies with the LDAP standards, which means that it can be used as a backbone, to unify directories that can only use LDAP.

Example: A Distributed Directory

The following diagram shows a single namespace that is served by five DSAs. The CA Directory DSAs communicate with each other using DSP, and they communicate with LDAP-only directory servers using LDAP.



When the client sends a query to the directory, the query is routed to the DSA that serves the relevant part of the namespace. The client does not need to know that the directory is distributed.

Distribution Protocols

In a distributed directory, the DSAs use the X.500 Directory Systems Protocol (DSP) to cooperate to resolve queries on any area of the namespace.

CA Directory fully supports the X.500 directory systems protocol, including the following:

Chaining

Performing queries through other DSAs.

Multichaining

Performing distributed searches across many DSAs.

Referrals

Informing clients where to find information.

How Distributed Searches Work (Multichaining)

Searches can span multiple DSAs.

A subtree search searches all entries below and including the base-object of the search. Some entries can reside on a different DSA from the base-object.

When you enable multichaining, the DSA searches for immediate subordinate DSAs after you find the base-object and then forwards the search to these DSAs. Results from all subordinate DSAs and the local DSA (the DSA containing the base-object) are collated before being returned.

How Routing Works

When a DSA in a backbone receives a request, it may need to forward (or *chain*) the request to another DSA to deal with the request.

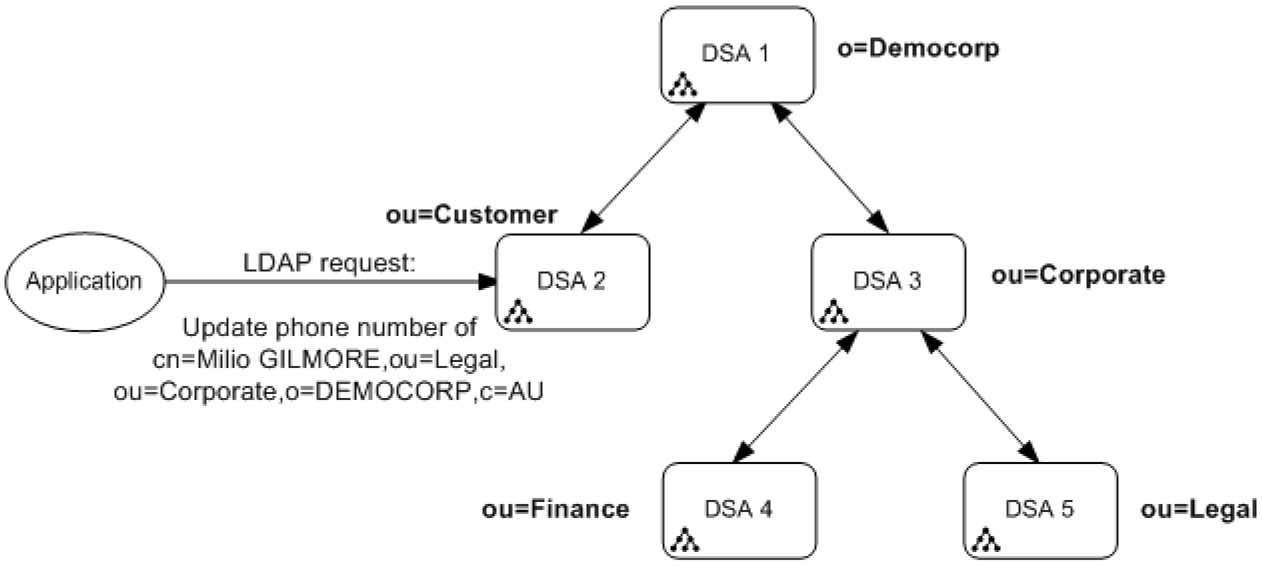
A request may be chained through a number of DSAs before finding the DSA capable of processing the request. Because the entire directory backbone shares the same configuration knowledge, a DSA receiving a request is able to forward that request directly to any other DSA within the domain.

A search request can require a subtree search spread over more than one DSA. In this case, the DSA containing the base object of the search performs the search locally and multi-chains the search to any DSAs that control part of the subtree to be searched. These DSAs return the search results back to the originating DSA, which collates them and returns the result back to the client.

Example: Routing an Update Request

The following diagram shows five DSAs that control a single namespace. Each DSA controls a portion of the namespace, represented by its prefix, and a relationship with the other DSAs in the domain.

In the following diagram, DSA 1 is the superior of DSA 2 and DSA 3. DSA 4 and DSA 5 are subordinates of DSA 3:



1. A client sends a request to DSA 2, to modify the phone number of the entry cn=Milio GILMORE,ou=Legal,ou=Corporate,o=DEMOCORP,c=AU.
2. DSA 2 cannot respond to the query, so it routes (forwards) the request to its superior, DSA 1.
3. DSA 1 also cannot respond, but because the entry is controlled by its subordinate, it routes the request to DSA 3.
4. DSA 3 routes the request to its subordinate, DSA 5.
5. DSA 5 modifies the email address, and sends a confirmation response back to DSA 3.
6. DSA 3 sends the response back to DSA 1.
7. DSA 1 sends the response back to DSA 2.
8. DSA 2 sends the response to the client.

Remote Operations

Each DSA has a context prefix that defines the base of the DIT controlled by that DSA.

When the DSA receives an incoming operation, it services the request locally when it is in the DSA's own section of the namespace. When the operation is not local, it chains the operation to a remote DSA except in the following circumstances:

- One (user) service control (either *chainingProhibited* or *localScope*) is set.
- The remote DSA is unavailable.
- You cannot establish a link of the appropriate authentication level to the remote DSA.
- There is no DSA knowledge for the area of the operation.

Depending on the circumstance, the remote operation returns one of the following options:

- Result
- Referral
- Service error
- Security error

See the X.500 standards for a complete specification of how distributed DSAs cooperate to resolve a query.

Caching Entries from Subordinate DSAs

Unlike DAP, LDAP does not have a LIST operation. So, when LDAP clients want to browse a directory, they invoke one-level searches that return object class only.

These one-level searches become a problem when there are many subordinate DSAs because a one-level search must be broadcast to each of the DSAs (whereas a LIST returns the names of the DSAs).

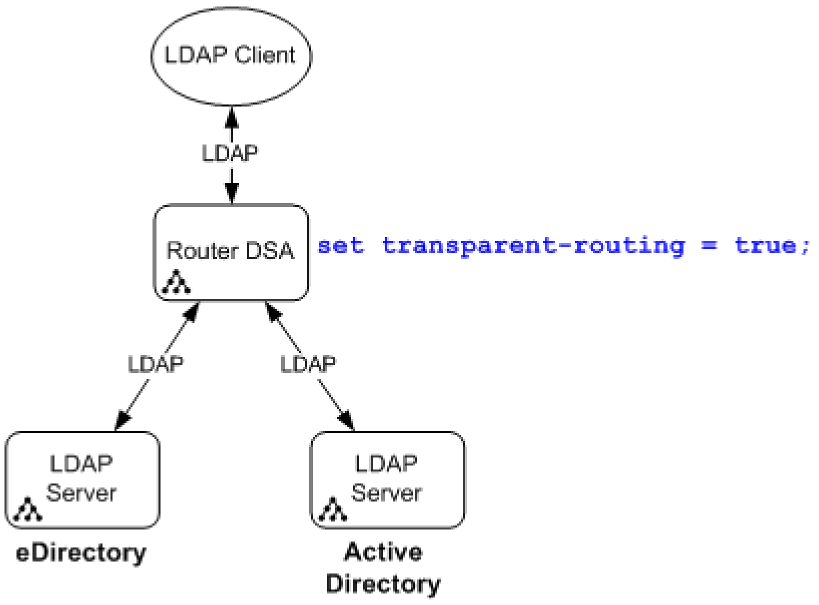
To stop the flooding effect that this creates, especially at first-level DSAs, the DSA caches the replies to one-level-search queries and makes them available to subsequent similar requests.

Only one-level searches returning object class are cached. These are the searches commonly used to browse the directory.

Transparent Routing

Transparent routing allows a router DSA to process LDAP requests and responses without requiring the controlling schema. This is useful if the router DSA is being used to link LDAP clients to an LDAP server, or the clients and server have schema that are not known by the router DSA. Transparent routing works with LDAP clients only.

By default, transparent routing is set to *false* as shown in the following diagram:



DSAs with the Same Prefix

Many DSAs can have the same prefix, and so serve the same namespace partition. A router DSA selects one of these DSAs to receive queries.

Having multiple DSAs with the same prefix improves the performance and availability of the directory service. These DSAs can be on the same computer, or on different computers.

You need to ensure that the DSAs are synchronized.

You can use groups of DSAs with the same prefix and the same data, as follows:

Failover

To improve the availability of the service, when one DSA fails, another automatically takes over its request load.

Load sharing

To improve performance, read requests are divided between two or more DSAs. This happens automatically between DSAs in the same site.

Load Sharing

Load sharing lets a router DSA distribute incoming requests evenly among all DSAs in the same site that serve the same namespace partition. This improves performance.

DXmanager automatically sets up load sharing between DSAs in the same site that serve the same namespace partition.

If you do not use DXmanager you need to set the DSA knowledge to enable load-sharing. The DSA flags you can set are as follows:

- load-share—This enables the DSA to take part in load sharing
- load-share-group—This specifies what load share group the DSA is in
- precedence—This specifies the order in which DSAs are allocated work.

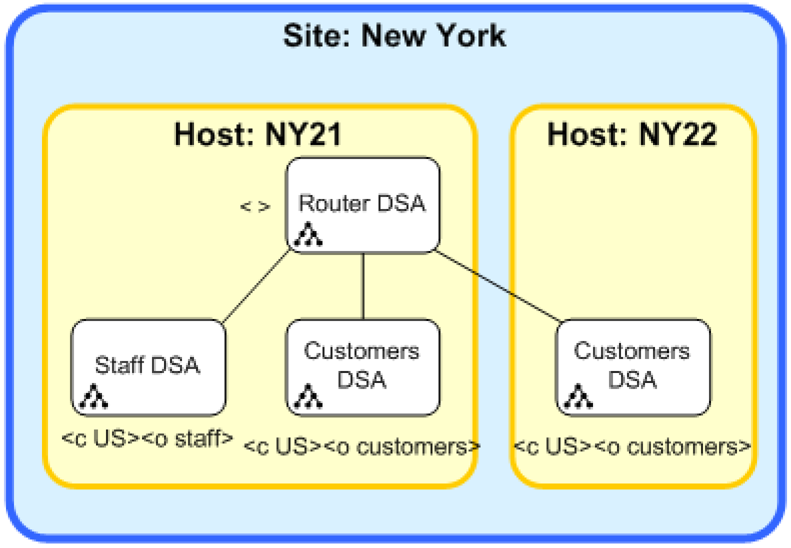
Load-sharing works best within a site.

The router DSAs use a *least-busy* load-sharing algorithm, which uses an ordered list of DSAs serving each namespace partition. The routers favor the DSAs highest in the list that are not busy.

Note: DXmanager requires that DSAs that serve the same namespace partition must be on different hosts. Also, DXmanager only allows load-sharing between DSAs in the same region as the router.

Example: Load-Sharing Within the New York Site

The following diagram shows a router DSA connected to three data DSAs within the same site:



The two Customers DSAs have the same prefix, so the router shares the load for that prefix between those two DSAs.

More information:

Sites and Load Sharing (see page 47)

Failover and Failback

Failover is the ability of a router DSA to continue to service queries even when a data DSA becomes unavailable. If the router detects that a DSA has failed, it resends outstanding requests to another DSA that serves the same partition, making the failure invisible to clients.

During normal operation, the standby DSA is kept synchronized with the primary data DSA in case it is needed.

Failover is important for systems requiring high availability and reliability.

More information:

Sites and Failover (see page 48)

Failover between Hosts

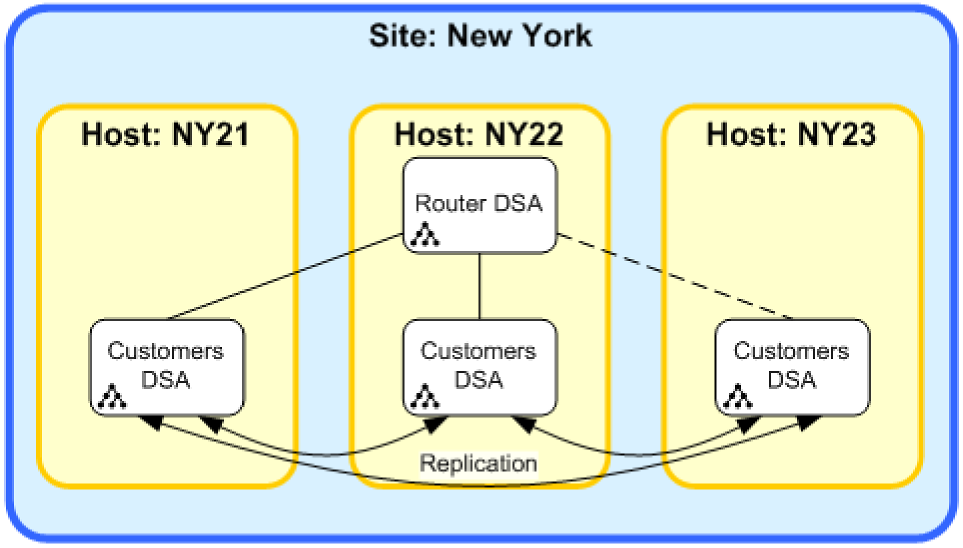
CA Directory automatically sets up failover between data DSAs in the same site that serve the same namespace partition. These DSAs are also automatically replicated.

Example: Failover to a Backup DSA

The following diagram shows a single router DSA that usually sends requests to the Customers DSAs on its own host, and on the NY21 host.

If both of these data DSAs fail, the router DSA automatically sends all requests to the Customers DSA on NY23. This change is invisible to the clients.

In this example, the Customers DSA on NY23 is a backup: it is used only if the other data DSAs fail.



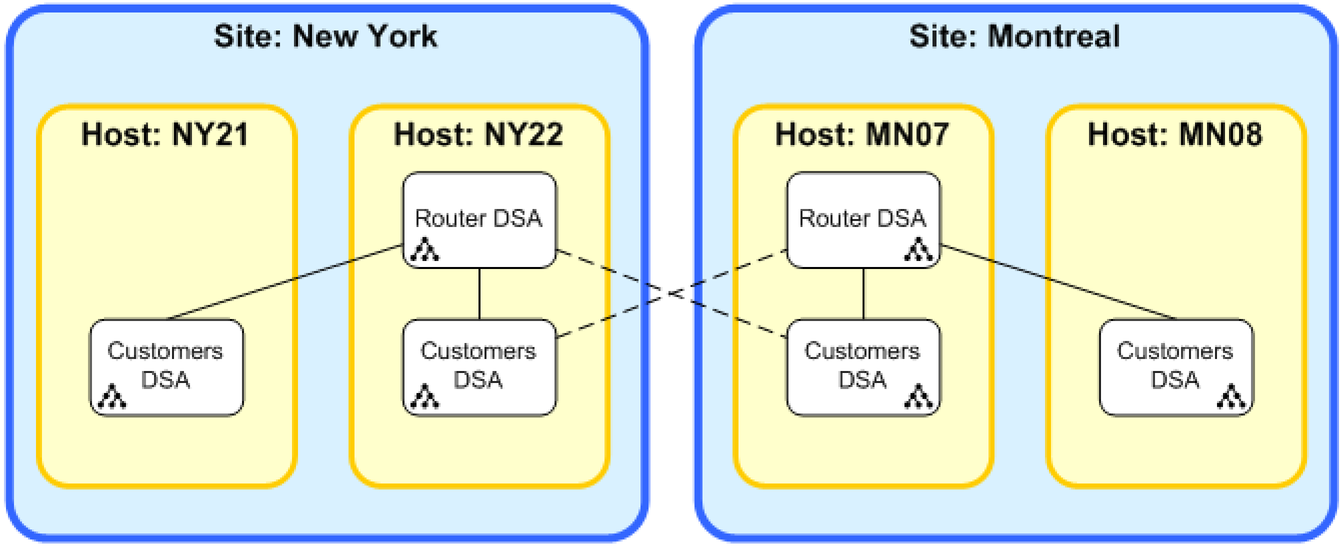
Failover between Sites

To improve performance, we recommend that each region have a router that directs queries to a local data DSA. The local data DSAs should be synchronized with each other.

If a local data DSA fails, each router directs queries to the remote data DSA. This maintains the availability and reliability of the directory.

Example: Failover between Two Sites

The following diagram shows an example directory backbone that is distributed across two sites, New York and Montreal.



The New York router cannot load-share to the Customers DSAs in Montreal because they are in a different site from the router.

However, if both of the New York Customers DSAs are unavailable, the router can failover to the Customers DSA in Montreal. Although the performance is lower until the New York DSAs are running again, the service is still available.

Note: Failover consistency is guaranteed within a region, but not between regions.

Client Failover between Router DSAs

If a router DSA fails, all access to the data DSAs stops. To prevent this, you can set up the client application to failover between router DSAs as follows:

1. Run more than one router DSA on more than one computer.
2. Configure the client applications to failover from one router DSA to the other.

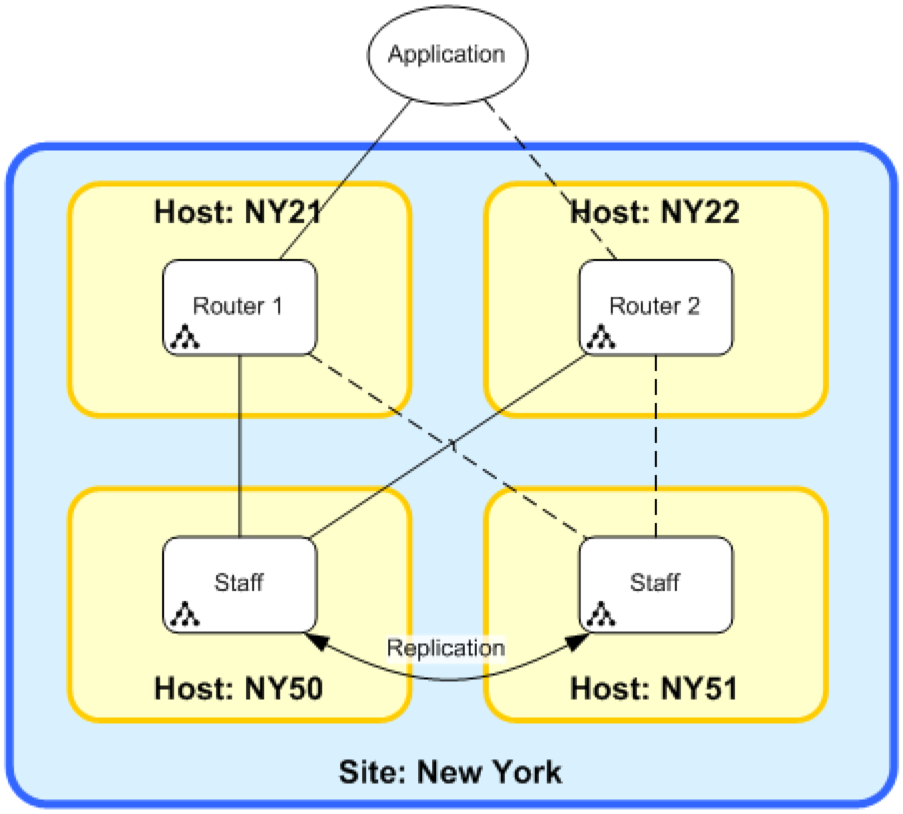
Nearly all large directory-enabled third-party applications can be configured to do this.

Example: Client Fails Over to Second Router

The following diagram shows a simple directory system with two kinds of failover.

As in Failover Between Hosts (see page 127), the router DSAs can fail over to the second data DSA. However, this system also shows that the client application can fail over between the two router DSAs.

These two router DSAs are set up to use the data DSAs in exactly the same way, which means that the service to the client application is not interrupted.



Other Methods for Router Failover

Other approaches have been taken, and each has its pros and cons as follows:

Round-robin DNS

This solution is easy to implement, but each client must maintain affinity with a single router DSA for the duration of its connection. Also the inherent delays in pushing out changes to the DNS records might cause a problem.

IP failover

Another good solution is to provide an immediate IP failover mechanism between the computers running router DSAs. There is some extra configuration required for a router DSA to bind to a new IP address, but it can be done. Ensure that the method you choose is immediate, does not require rebooting, and handles failback.

Network load balancers

These are not recommended. Network load-balancers may reduce the performance of CA Directory.

Failback

Failback is the restoration to normal service of a CA Directory DSA after failing over and recovering.

When a DSA fails, failover operation comes into play. It redirects all requests for the primary DSA from the primary DSA to the secondary DSA. When, later, the primary DSA becomes available again, CA Directory automatically restores the normal situation and requests for the primary DSA are again sent to the primary DSA. CA Directory tries to restore normal operation every five minutes during failover operation.

Security in Distribution

To set up a DSA in a distributed system, you should be aware of how security works in a distributed system.

Remote Bindings

The DSA dynamically binds to and unbinds from remote DSAs. A DSA maintains at most one binding per security level (set by the auth-levels list in the DSA definition) to a remote DSA. When a DSA has an established DSP binding of the correct security level to a remote DSA, it uses this binding. A second binding of the same security level is not set up.

The DSA definition supports trust flags that enable a DSA to upgrade or downgrade a link between DSAs. For example, when a link between two DSAs supports only anonymous connections, a credentialed user can access the link when the receiving DSA supports the *allow-downgrading* trust flag. Conversely, when the only allowed DSP link is a clear-password link, an anonymous user can access the link only when there is support for the *allow-upgrading* trust flag.

A DSP binding to a remote DSA unbinds after the dsp-idle-time is exceeded. Ensure that you set this value higher on router DSAs than on data DSAs.

Incoming DSP Credentials

When a local DSA receives a bind with credentials from a remote DSA, it checks the credentials against a matching (remote) DSA configuration. When you do not configure a relevant remote DSA, the system rejects the incoming bind.

More information:

How to Encrypt Communication to DSAs (see page 183)

Outgoing DSP Credentials

If a remote DSA requires authentication, the local DSA must send credentials when binding to the remote DSA. To be able to do this, the local DSA must have credentials (user name and password) defined in its own *set dsa* definition.

Connect to Other LDAP Servers

CA Directory supports Lightweight Directory Access Protocol (LDAP), which allows and any LDAP-conformant client to access CA Directory.

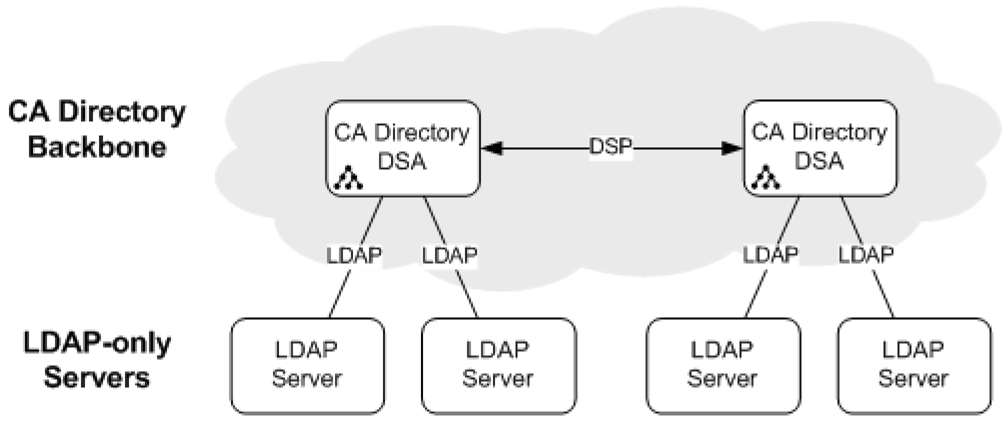
LDAP is fully integrated with CA Directory and provides greater performance and efficiency than gateway approaches. Another advantage of integrated LDAP is that it obeys the DSA schema. This means that you configure a new schema only once, because both LDAP and DAP protocols share the same configuration.

CA Directory also supports the integration of LDAP servers into a distributed directory backbone. CA Directory can send requests to one or more LDAP servers and process the results returned from them as if they were normal X.500 directory hosts. This enables LDAP servers to be integrated into a fully distributed directory framework.

LDAP Integration with CA Directory

LDAP servers do not support the DSP protocol. This means that you cannot perform a distributed search over a number of LDAP directory servers. To solve this problem, CA Directory provides a facility called DXlink, which lets you integrate LDAP servers into a fully distributed directory backbone and handle requests as if they were X.500 directory servers.

The following diagram shows this:



Connect to an LDAP Server

In DXmanager, you can deploy a namespace partition to an LDAP server. This means that DXmanager knows that the LDAP server is a part of the directory backbone.

Although DXmanager cannot start, stop, or reconfigure LDAP servers, it can monitor them.

User Credentials on DXlink Binds

LDAP servers expect connections only from LDAP users; therefore, DXlink must make the X.500 backbone look like an ordinary LDAP user.

A complication arises with name and password security (simple credentials). In DSP, a single link between DSAs can support any number of users, because user information is passed with each DSP request. However, in LDAP, links cannot be shared, so the CA Directory DSA must set up separate links for every LDAP user.

When the DSA is acting as a direct pass-through from a user to an LDAP server and the user's name is on the LDAP server, the DSA sets up a separate link for that user and uses their credentials in that link.

Setting User Credentials for LDAP Operations

If either of the following is true, you can set the credentials used in the DXlink connections in the LDAP server configuration file:

- The user invoking the request is authenticated using a DN that is outside the LDAP server.
- More than one DSA is in the path to the LDAP server.

  For example:

  ```
  set dsa LDAP1 = {
      ...
      ldap-dsa-name   = <c US><o “Ace Industry”><cn “Fred Smith”>
      ldap-dsa-password = fredspassword
      ...
  };
  ```

The LDAP DSA name must be a valid entry in the LDAP server because all requests from the backbone use the permissions that are granted to this entry.

The DSA in the previous example expects credentials to be returned on the bind confirm sent by the LDAP server. If no credentials are returned, then the bind is rejected.

The knowledge reference of the LDAP server can include the trust flag *no-server-credentials*, which indicates to the DSA that the LDAP server will not return credentials on a bind.

When this flag is set, then the DSA accepts a bind confirm result returned from the LDAP server if it does not include credentials, as in the following example:

```
set dsa LDAP1 = {
    ...
    trust-flags = no-server-credentials
    ...
 };
```

Automatically Authorizing LDAP Operations

When a directory backbone performs operations over DXlink, some operations on the target LDAP server may require that the user be authorized for that operation.

You can include the *dsp-ldap-proxy* link flag in the DXlink knowledge to cause the last DSA in the chain to use the authorization of the originating user to perform operations on the LDAP server.

Important! This may compromise security because the originating user is never authenticated by the LDAP server.

Usually, the last DSA in the chain binds to the LDAP server using the credentials specified in the *ldap-dsa-name* and *ldap-dsa-password* flags.

If the *dsp-ldap-proxy* flag is also set, then the DN of the user that made the initial bind is added to the following subsequent requests:

- search
- compare
- modify
- add
- delete
- modify DN

If the initial bind was anonymous, no DN is added to subsequent requests.

To do this, the DSA that chains the operation over DXlink adds the originator DN to a LDAP proxy control on the request. The LDAP server must permit the entry in the LDAP DSA name the authority to proxy all users.

Note: The *dsp-ldap-proxy* link flag can only be used if the target LDAP server supports the LDAP Proxy Authorization control.

Prefix Mapping

Prefix mapping lets LDAP servers, other DSAs, or agents map into a backbone namespace.

Prefix mapping is useful for collecting disjointed subtrees under a common node. Applications include transient groupings (such as task forces) or logical groupings (such as libraries where individual libraries are spread across an organization or location tree).

Prefix mapping is also useful for if you want to integrate an LDAP server into a backbone. Because LDAP servers have no concept of distribution, their DITs usually contain the first- or second-level nodes. This makes it hard to integrate them into a host DIT.

To enable prefix mapping, configure the DSA's native prefix, using the set dsa command or DXmanager.

Example: Prefix Mapping

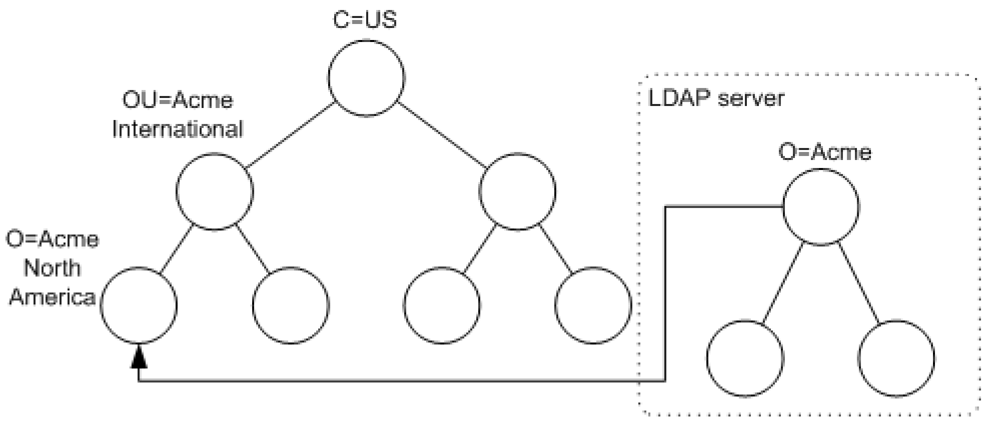
In this example, a CA Directory backbone controls the nodes "c=US" and "o=Acme". A DSA has the knowledge with the following prefix definition:

```
prefix = <o us>
```

There is also an LDAP server, which controls the partition “o=Acme, c=US”. Originally this controlled the North American part of Acme but must now be incorporated into the backbone. To do this, the knowledge for the LDAP server must contain the following prefix and native-prefix definitions:

```
prefix=<o US> <O Acme International> <o Acme North America>
```

```
native-prefix = <o Acme>
```



The result is as follows:

1. The backbone DSA receives a request for the following:

   ```
   "ou=Acme North America, o=Acme International, c=US"
   ```

2. It replaces the prefix it received with the LDAP server's native prefix:

   ```
   "o=acme"
   ```

3. It forwards the request to the LDAP server,
4. The LDAP server sends back its results
5. The backbone DSA replaces the LDAP server's native prefix with the original prefix and sends the results back to the client.

Horizontal Partitioning

If you have a large flat namespace, you can improve performance by partitioning the namespace, so that different DSAs each serve different parts of the same level of namespace. In CA Directory, *horizontal partitioning* is a method of doing this.

A horizontal partitioning setup consists of a router DSA and a number of data DSAs.

The router and its data DSAs serve the namespace. The clients do not know which data DSA stores the data for each part of the namespace.

The namespace data is partitioned among the data DSAs by hashing the RDN of each entry. Each data DSA is distinguished by the hash value assigned to the prefix. Apart from the hash value, the prefix specification is identical for all the DSAs in the set.

Limitations in Horizontal Partitioning

Because horizontal partitioning uses the hash of the RDN, you cannot rename horizontal partitioned entries. You should delete the old entry and create a new one.

Partitions cannot have subordinate DSAs.

In a horizontal partition, a multi-chained search gives a separate dxEntryCount for each DSA which returns a hit. To get a consolidated number, use the value in dxTotalEntryCount.

Specify Horizontal Partitioning

Specify horizontal partitioning to partition a namespace amongst DSAs.

To specify horizontal partitioning

1. Select the attribute whose value you will use to partition the namespace.
2. Select a hash algorithm

   You can choose *hash1* or *hash2.*
3. For each DSA in the partitioning, append to the prefix the string that determines the partition hashing.

   You specify the prefix in the knowledge, so use either DXmanager or the set dsa command (prefix parameter).
4. Ensure each data DSA has *disable-client-binds* set to true.

   This forces requests to the DSAs to go through the router.

Example: Configuration for Horizontal Partition

The configuration of the DSAs is set up as follows:

- The router has prefix: <c AU> <ou Users>
- DSA_A has prefix: <c AU> <ou Users> <guid "[hash1(3)=0]" >
- DSA_B has prefix: <c AU> <ou Users> <guid "[hash1(3)=1]" >
- DSA_C has prefix: <c AU> <ou Users> <guid "[hash1(3)=2]" >

From a client point of view all entries appear in the namespace <c Au> <ou Users>.

When the router receives a request, the router applies the specified hash function (hash1) to the specified attribute value (guid) in the request. If the result is zero, it routes the request to DSA_A. If the result is one, it routes the request to DSA_B, and so on.

Chapter 7: Set Up Authentication

This section contains the following topics:

Authentication Levels

Authentication is the process of validating users. During authentication, the server asks itself, “Is the user who he or she says they are?”

Each DSA has one or more *authentication levels*. The authentication levels assigned to a DSA define what credentials a user must present to bind to and query that DSA.

CA Directory supports three levels of authentication:

- Anonymous authentication (see page 140)
- Clear-password authentication (see page 141)
- SSL authentication (see page 142)

Set the Authentication Level

You can set the authentication level at the following levels in DXmanager:

- Backbone—This works as a default value for the whole directory.
- Namespace partition—This affects all DSAs that serve that partition.

To change the authentication levels

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. To change this value for the whole backbone, do the following:
    a. Click the Maps tab and display the Topology map.
    b. Right-click on the Backbone icon, and then select Edit properties.
    c. In the Security section, check or uncheck one or more of the Authentication levels boxes, and then click OK.
4. To change this value for a single namespace partition, do the following:
    a. Click the Maps tab and display the Namespace map.
    b. Right-click on the partition you want this setting applied to, and then select Edit properties.

       The inherited authentication levels are displayed in the Security section.

       If you want to change the value, click Override, and check or uncheck one or more of the Authentication levels boxes.

       If you want to inherit the value from the backbone, click Inherit.
    c. Click OK.

Note: You do not need to set the level of authentication for an incoming request to a DSA, because the DSA does this automatically.

Anonymous Authentication

Anonymous authentication lets users connect to a directory without providing credentials.

This is useful for public directory services, because user identification is usually not important.

Clear-Password Authentication

Clear-password authentication (sometimes called *simple* authentication) allows users to connect or bind to a directory by providing a username and password.

The following conditions are required for clear-password authentication to work:

- The name corresponds to a real entry in the directory.
- That entry has a password attribute.
- The user supplies a username and password.
- The minimum authentication of all the DSAs must not include the value *ssl-auth*.

Authentication fails and the bind is refused in the following cases:

- The entry named by the user cannot be found.
- The entry named by the user name does not contain a password attribute.
- The password provided does not match the password value of the attribute in the entry named by the user name.

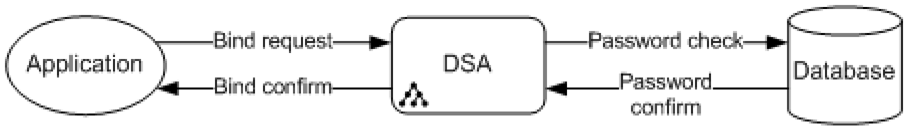
How a Connection Is Established with Clear-Password Authentication

Clear-password authentication consists of sending the DSA the DN of the client, and the client's password.

The diagram below illustrates the following steps:

1. The client sends a bind request with its username and password.
2. The DSA checks the username and password against the relevant DN entry.
3. If the username exists and the password matches, the bind is authenticated and established.

   If the username does not exist, or the password is incorrect, an BIND REFUSE message is returned to the client.



Note: Once a bind or connection is established, all further client operations or directory requests are subject to access control rules.

SSL Authentication

Strong authentication uses SSL certificates to protect LDAP and X.500 access by encrypting data with Secure Sockets Layer (SSL) security. When certificate-based authentication is used, all communication on the binding set up by the bind use SSL encryption.

SSL certificate based authentication is typically used in environments where personal or company data requires protection, for example, an online banking environment.

SSL authentication has two parts:

- The SSL connection
- The directory connection (using a bind)

Two variants are allowed:

- Simple SSL
- Authenticated SSL

Simple SSL authenticates the server only, while Authenticated SSL authenticates both the client and the server.

How an SSL Connection Is Established

An SSL connection always starts with an exchange of messages between the client and the DSA server, and is commonly referred to as the SSL handshake. The handshake allows both the DSA and client to authenticate themselves using public-key techniques (trusted certificates).

The following diagram illustrates the first part of the SSL certificate-based authentication process:

1. The client sends a bind request, including a certificate.
2. DSA validates the connection request by checking the validity dates and checking the issuer of the certificate against the configured trusted roots.
3. If the certificate details are correct, the DSA establishes an SSL connection with the client application.

How a Directory Connection Is Established

After an SSL connection has been established between a client and a DSA, the client can use that connection to request a bind to the directory.

In LDAP, this is known as SASL/EXTERNAL.

In a distributed or X.500 environment, the bind external procedure is used. This tells the directory to use the certificate from the link layer.

The directory connection is established over an existing SSL link as follows:

1. The client sends a bind request to the directory.
2. The DSA checks the directory entry named by the subject DN contained in the certificate.
3. If the DN named in the subject DN of the certificate match those in the directory, then the DSA accepts the bind request.

Note: In a secure environment, you can choose to bypass the DSA check on the DN.

Bypass the Entry Check

Usually, during SSL authentication, the DSA verifies that the entry exists. To bypass this entry check, add the following command to the DSA's configuration:

```
set ssl-auth-bypass-entry-check = true;
```

When this is set, while authenticating the client, the DSA does not check that an entry with a distinguished name matching the subject field in the certificate of the client exists in the directory.

Configure Distributed User Authentication

Authentication in a distributed environment refers to the behavior of user authentication in a multi-DSA environment.

In a distributed system, the namespace is divided into many DSAs. This means that users could bind to one DSA and request information on another. Users can also bind to one DSA, which then forwards the user's credentials to a second DSA to confirm the bind.

How the networked system of DSAs responds to these requests depends on how they are configured.

To configure authentication in a distributed environment, follow these steps:

1. Set appropriate authentication levels for each DSA (see page 140).
2. (Optional) Configure relevant DSAs to allow forwarding of password checks (see page 144).
3. (Optional) Configure all relevant DSAs to trust conveyed originators (see page 147).
4. Check the authentication link between all DSAs (see page 148).
5. Set up DSP and DISP requirements (see page 150).

Bind Requests in a Distributed Environment

DXmanager lets you define how much each DSA should trust another. By default, security is tight. The settings let you selectively relax security between DSAs.

In a distributed network of DSAs, users can bind to one DSA when their entries are held on a second DSA. When the initial bind is made, the DSA can forward the password compare check to a second DSA if certain authentication parameters are set.

To allow users to bind to a local DSA when their details are held on a remote DSA, the *Allow check password* value must be set to *true* in DXmanager. This is set to *true* by default.

When a bind is requested, the local DSA forwards a Password Compare request to the remote DSA. If this returns a Compare Confirm with the assertion *true*, the local DSA returns a Bind Confirm message to the user.

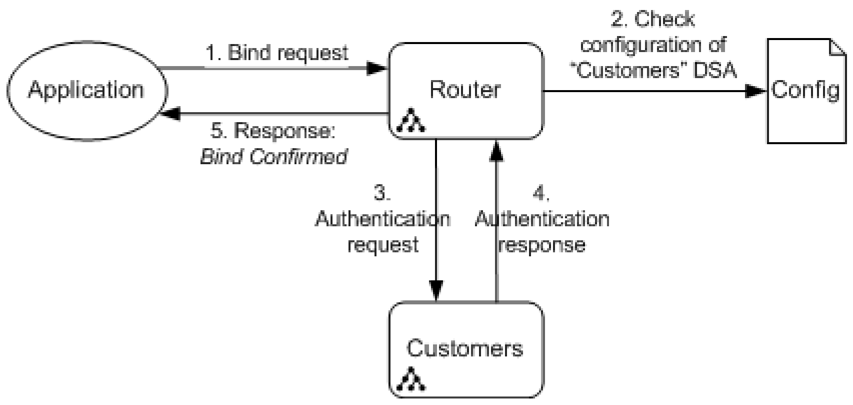
A request can include a *chaining-prohibited* control. CA Directory ignores this control.

Example: Forward a Password Check to Another DSA

In this example, the router DSA contains no entries. This means that the router DSA must delegate checks of user credentials to another DSA.

The Customers DSA contains the entries for the customers, including the credentials required during binds. The *Allow check password* setting for the Customers DSA is *true*.

The following diagram shows how the router DSA delegates a password check to the Customers DSA:


Application
1. Bind request
5. Response:
Bind Confirmed
Router
2. Check configuration of "Customers" DSA
Config
3. Authentication request
4. Authentication response
Customers

1. The router DSA receives a bind request from a user whose credentials are stored in the Customers DSA. The router DSA cannot check the user's credentials, but it knows that the Customers DSA can.
2. The router DSA checks the configuration of the Customers DSA to see whether it can trust the Customers DSA to authenticate a user. The *Allow check password* setting indicates that this is permissible.
3. The router DSA requests the Customers DSA to authenticate the user.
4. The Customers DSA responds with the user's authentication.
5. The router DSA returns the bind confirmation to the client.

Change the Allow check password Setting

The Allow check password setting allows a DSA to delegate the password check to another DSA.

You can set the value for this configuration item at the following levels:

- Backbone—This works as a default value for the whole directory.
- Namespace partition—This affects all DSAs that serve that partition.

To change the Allow check password setting

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. To change this value for the whole backbone, do the following:
   a. Click the Maps tab and display the Topology map.
   b. Right-click on the Backbone icon, and then select Edit properties.
   c. In the Security section, click the Show advanced link.
   d. Check or uncheck the Allow check password box.
4. To change this value for a single namespace partition, do the following:
   a. Click the Maps tab and display the Namespace map.
   b. Right-click on the partition you want this setting applied to, and then select Edit properties.
   c. In the Security section, click the Show advanced link.

      The inherited value for this setting is displayed here.

      If the value is set at this level and you want to make it inherited, click Inherit.

      If the value is inherited and you want to change it, click Override, and then check or uncheck the Allow check password box.
5. Click OK.

How User Authentication Is Conveyed between DSAs

In a networked system of DSAs, a user can bind to a DSA, and then request information that is held on another DSA. You can use the *Trust conveyed originator* option to permit the first DSA to convey the user's authentication to the second DSA.

The following steps provide a high-level overview of how user authentication is conveyed between DSAs:

1. A user binds to a DSA. The bind request includes the user's DN, and the user's credentials.
2. The DSA authenticates the user.
3. The user makes another request that the current DSA cannot fulfill.
4. The DSA passes the request to another DSA that can fulfill the request. The request includes the user's DN and authentication.
5. The receiving DSA must decide whether to trust the user's authentication. To do this, it looks at the configuration of the first DSA for the *Trust conveyed originator* option.
6. If the receiving DSA finds the *Trust conveyed originator* option, it accepts the request. Even though the user was authenticated on the first DSA, it is treated as if it had been authenticated on the second.
7. The receiving DSA uses the DN of the originating user to determine what access controls to apply to the request.

Example: Convey User Authentication

In this example, a client is connecting to one DSA and requesting information from a second DSA. Rather than repeat user authentication, the UNSPSC DSA can be configured to trust all user authentication being passed by the Democorp DSA.

To do this, the UNSPSC DSA's configuration includes the *Trust conveyed originator* option.

How the Authentication Link Is Conveyed between DSAs

Traffic on any link is generally carried at the same security level.

If the initial bind was made using a strong SSL certificate-based connection then communication between DSAs occurs at the SSL security level. Alternatively, if the initial bind was made using no authentication, then all communication would occur at the same level.

Strong security

SSL authentication is carried on an SSL bind

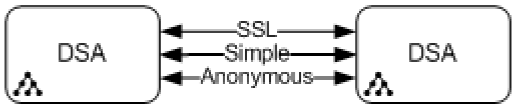
Simple security

Simple authentication is carried on a clear-password bind

No security

Anonymous authentication is carried on an anonymous bind

The following diagram illustrates this:



This presents two potential issues:

- A user may bind to one DSA using simple authentication, and then request information from a second DSA that requires SSL authentication.
- A user may connect to one DSA using SSL authentication but then is refused any distributed operations because further DSAs do not support SSL. Both issues result in the following error message:

  ```
  No compatible link type.
  ```

To overcome these potential issues, you can either change the authentication levels so that a compatible link can be established, or upgrade or downgrade the trust levels between distributed DSAs.

A link is upgraded if a DSA uses a higher level of authentication to forward a clients request to another DSA that is higher than the authentication level used by the client to bind to the DSA.

Upgrade the Link Type

If an anonymous user sends a request but a remote DSA can only provide authenticated links, the user is denied distributed operations, and receives the error message “No compatible link type.”

To upgrade the link type between DSAs, set the following command in the appropriate knowledge configuration file:

```
trust-flags = allow-upgrading
```

Important! Allow upgrading presents a serious security risk. For example, you are allowing an anonymous user to act as if they had presented a username and password.

Example: Upgrade the Link Type

A user is connecting to one DSA using anonymous authentication and then requesting information from a second DSA that requires simple authentication (username and password). Unless the link is upgraded, the user receives the error message *No compatible link type*.

To upgrade the link, the DemoCorp DSA knowledge file includes configuration commands as in the following extract:

```
set dsa UNSPSC =
...
trust-flags=allow-upgrading
...
```

Downgrade the Link Type

If an anonymous user sends a request but a remote DSA can provide only SSL or password-authorized links, the user is denied distributed operations, and receives the following error message:

```
No compatible link type.
```

To downgrade the link type between DSAs, set the following command in the appropriate knowledge configuration file:

```
trust-flags = allow-downgrading
```

Important! You should only use this command to permit unauthenticated DSA links within the one organization (or on the one host), or to grant access to a public server.

Example: Downgrade the Link Type

In this example, a user is connecting to one DSA using SSL authentication, and then requesting information from a second DSA that requires simple authentication (username and password). Unless the link is downgraded, the user receives the error message *No compatible link type*.

To downgrade the link, the Democorp DSA is configured with the trust flag *allow-downgrading*.

DSP Authentication

Directory system protocol (DSP) is an X.500 exchange protocol between DSAs. It supports anonymous, simple and SSL authentication. The following sections summarize DSP simple and SSL authentication.

DSP Simple Authentication

DSP binds are subject to the *min-auth* setting. This means that if the local DSA has enabled authentication, an incoming DSP bind request must have valid bind credentials.

Authenticated DSP binds use mutual authentication. During the bind process, each DSA supplies its own credentials to the other DSA, and each DSA also checks the credentials supplied to it by the other DSA.

Credentials Used during DSA Binds

An incoming bind from another DSA is accepted only when there is a DSA knowledge configuration that meets all of the following conditions:

- The distinguished name of the credentials matches the remote DSA name.
- The password in the credentials matches the remote DSA password.
- The address of the binding DSA matches the address of the remote DSA.

We recommend that the DSAs in a DSP-meshed network share the same group knowledge configuration file so that all the combinations of names and passwords are known to each DSA.

DSP SSL Authentication

To enable SSL authentication between DSAs, the *auth-levels* flag of the receiving DSA must include the value *ssl-auth*.

If you want only DSA-to-DSA connections to be encrypted, SSL authentication is unnecessary. Instead, set the *link-flags* element in the DSA knowledge to *ssl-encryption*.

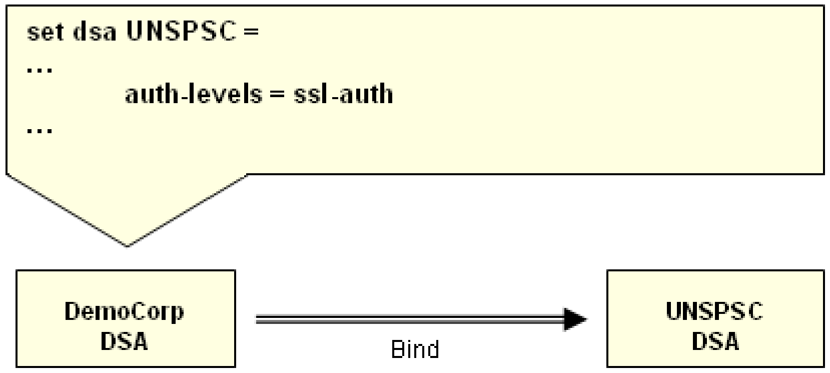
How DSP SSL Authentication Works

This section describes the DSP SSL authentication process.

1. The DSA sending the bind request checks its own knowledge file to determine if it can use SSL.
2. The sending DSA then checks the knowledge file of the receiving DSA to see if it can accept an SSL connection. If it cannot, the bind attempt is aborted. If it can, the DSA sends the SSL bind request.

Example: DSP SSL Authentication

In this example, the Democorp DSA finds the *ssl-auth* flag in its copy of the UNSPSC knowledge file, so it sends the SSL bind request as follows:



```
set dsa UNSPSC =
...
        auth-levels = ssl-auth
...
```

Password Storage

To check a password, an application requests a compare operation. The DSA then hashes (encrypts) the candidate password and compares it with the value saved in the directory.

Activate Secure Proxy

To activate proxy access, enter the following command:

```
set trust-sasl-proxy = dn, …
```

In this command, *dn* is the distinguished name of a trusted proxy. The proxy must bind using SASL/EXTERNAL over SSL.

Chapter 8: Set Up Access Controls

This section contains the following topics:

Access Controls

The purpose of access controls is to protect data from unauthorized access or modification. After a client connects to a directory, access controls are used to determine which data and operations that user can access.

CA Directory access controls are based on the 1993 X.500 standards. However, the 1993 X.500 access controls are difficult to understand and work with. CA Directory provides access control rules that are easier to work with, and maps these rules to the X.500 access controls.

How Access Controls Work

Before performing any type of operation, an access control asks the following question:

- Is this *client* permitted to perform this *operation*, on this d*ata*, in this *subtree*, at this *time*?

  Where:

  - A *client* is a user, a role, a group of users, or a subtree of users
  - An *operation* is one of the usual directory services, including compare, list, search, read, add, remove, modify, or modify-dn
  - The *data* can be an entry or attribute
  - A *subtree* is a particular area of the namespace
  - The *time* is a particular minute of the day or a particular day of the week

When the answer to this question is *yes*, the operation can proceed. When the answer is *no*, the operation is refused.

For information about how aliases work with access controls, see Access Controls and Aliases (see page 395).

Access Control Policies

Each directory backbone has one access control policy. This policy is the collection of rules that define who can access what, and under what circumstances. We recommend that you apply the same access control rules to every DSA in the backbone.

Although only one access control policy can be in place at any given time, the policy can have different effects at different times. For example, a policy can contain one set of access controls for business hours and another for after hours.

Access Control Rules

To define your access control policy, you define a set of rules.

Each access control rule determines the operations that can be performed by a user or client. A set of rules defines the complete access control policy.

For example, you can create rules that specify the following:

- Define access privileges for super users and public users
- Allow authenticated users to update their own entries
- Protect parts of the directory from public view

All access control rules contain parameters that specify who, where, and what the rule applies to. Rules are also cumulative, which means you can define a very complex access control policy.

Rule Configuration

Access control rules are stored in one or more configuration files. To change a rule you need to edit the file containing the rule. For the rule to take effect, the DSAs that source the access control file must be restarted after the file has been edited.

Where Access Control Rules Should Be Stored

Include the commands that define the access control rules in a configuration file, which should be a .dxc file in the DXHOME\config\access directory.

You can also group access control files into a group configuration file (.dxg). This helps you share policies among a number of DSAs.

Ensure that each DSA's initialization file sources the file that contains these commands. The DSA then activates these commands when it starts up or is reinitialized.

Limitations on Using Access Control Rules

The main limitations on using Access control rules are as follows:

- Access control rules are defined on named entries, subtrees or attributes. If you rename these items, then the access control rule no longer protects those items.
- If you deny access to a naming attribute and do not deny access to entries lower down in the tree, then the naming attribute is still visible.
- Access controls are effective only when you enable access controls.

Access Levels

Static access control rules have a hierarchy of precedence. When a DSA receives a request, the DSA tests whether to accept the request by using the applicable access control rule with the highest precedence, and ignores any other access control rules. The precedence levels are as follows:

1. (Highest precedence) Super user access level.

   Rules at this level grant access rights that cannot be taken away.

2. Administrative user access level.

   Rules at this level grant access rights that cannot be taken away.

3. Protected items access level.

   Rules at this level deny access rights that have been given at a lower precedence.

4. Registered user access level.

   Rules at this level grant access rights, but the rights can be taken away.

5. (Lowest precedence) Public user (anonymous) access levels.

   Rules at this level grant access rights, but the rights can be taken away.

Example: Access Levels with Multiple Rules

Assume that access controls are set on, and that an item has the following access control rules (and that no other rules are set):

- Administrative user access level: modify permission only is specified

  This means that administrative users can perform modify operations. (This implicitly means that they can also perform read permissions.)

- Protected item access level: all permissions are specified

  This means all operations are denied.

- Registered user access level: modify permission only is specified

  This means that registered users can perform modify operations. (This implicitly means that they can also perform read permissions.)

For that item, within the scope of those rules, and for the users specified in those rules:

- The item is invisible to public users.

  No access rule is specified, so public users have no access.

- The item is invisible to registered users.

  The registered users access level rule allows registered users to read and modify this item, but because the protected items access rule is a higher precedence, the DSA uses that rule (which is a denial) and ignores the registered users rule.

- Administrative users can modify and read the item.

  The administrative users access level rule allows read and modify operations, and this rule has a higher precedence than the protected items rule.

- Since there is no super user access rule, there are no super users.

Example: Access Levels with Multiple Rules and Specific Permissions on Protected Items

Assume that access controls are set on, and that an item has the following access control rules (and that no other rules are set):

- Administrative user access level: modify permissions only are specified.

  This means that administrative users can perform modify operations. (This implicitly means that they can also perform read permissions.)

- Protected item access level: modify permissions only are specified.

  This means modify permissions are not allowed, but all other operations are allowed.

- Registered user access level: modify permissions only are specified

  This means that registered users can perform modify operations. (This implicitly means that they can also perform read permissions.)

For that item, within the scope of those rules, and for the users specified in those rules:

- The item is invisible to the public users.

  No access rule is specified, so public users have no access.

- Registered users can read the item.

  The registered users access level rule allows both modify and read operations, but the protected items rule disallows the modify operations.

- Administrative users can modify and read.

  The administrative users access level rule has a higher precedence than any protected items rule.

- Since there is no super user access rule, there are no super users.

Super User Access

Super user access is identical to administrative user access that has the following options:

- Permissions are set to allow all.
- Entry level access control is specified. That is, no attributes are specified.
- Scope is set to be the whole directory.

More information:

Create Access Control Rules (see page 168)

Administrative User Access

Administrative user access gives users access rights over specified scopes. These scopes can be a subtree, an entry, or designated attributes in an entry or a subtree.

You can give a user Administrative access to only some attributes in a subtree. Users then cannot add or remove entries; they can only work with the selected attributes.

Administrative user access has a higher precedence than protected items access level.

More information:

Create Access Control Rules (see page 168)

Protected Items

The *set protected-items* command totally or partially denies access by public and registered users to a subtree, entry, or attributes in a subtree or entry.

For example, you might give registered users read access to the whole directory, and then specify personnel records as protected items. This makes the personnel records invisible to registered and public users.

Protected items access rules never grant an access. They specify what sorts of access are denied.

Protected items access rules take precedence over public users and registered users access level rules.

Protecting Entries and Subtrees

You can protect entries and protect subtrees. These cause slightly different outcomes.

Protect an entry

When you protect an entry, neither registered nor public users can read it. It can, however, appear in a list result if the user can access the parent.

This occurs because the protected entry may have subordinates that are visible to the user.

Protect a subtree

If the same entry has protection as a subtree, a registered or public user cannot read it, and it does not appear in a list result as a subordinate entry.

Protecting Passwords

Protecting a password makes the password attribute invisible to public and registered users.

Because authentication requires the name of an entry, and that entry must contain a password, hiding the password makes login entries indistinguishable from other entries.

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: Let a User Modify All Attributes in Their Own Entry Except "role"

In this example, the *set reg-user* rule gives the user 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).

Registered Users

Registered user access gives registered (authenticated) users access rights over specified scopes. These scopes can be a subtree, entry, or designated attributes in an entry or subtree.

You can give a registered user access to only some attributes in a subtree. The user then cannot add or remove entries: they can only work with the selected attributes.

Registered user access level rules are a lower precedence than protected items access level rules.

More information:

Create Access Control Rules (see page 168)

Public Users

Public user access gives all anonymous users access rights over specified scopes. These scopes can be a subtree, entry, or designated attributes in an entry or subtree.

Any access right you give for public users is given to all users, unless the access is taken away by a protected items access rule. Public user access level is the lowest precedence access level.

More information:

Create Access Control Rules (see page 168)

Example Access Control Policy

The directory architect has created the following plain-English access control policy:

- DSA administrators have all privileges.
- Authenticated users (that is, all users except public users) can update their own entries.
- PABX operators can update all telephone numbers.
- Public (anonymous) users can view, but not change, the name, email address, and telephone number of any staff member, and can also view, but not change, anything in the *public* subtree.
- Passwords must be invisible to everyone except DSA administrators.

Set Up the Example Access Control Policy

The directory architect has created a plain-English access control policy. To embody the policy in access control rules you need a set of configuration commands.

To set up the Example Access Control Policy

1. Enable access controls:

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

   This denies all users any access. This provides a clear basis on which to apply layers of permissions.

2. Delete any existing access control policies:

   ```
   clear access;
   ```

3. Give DSA administrators all privileges (assuming a role called DSA-administrators has been defined):

   ```
   set super-user DSA-Administrators {
       role=DSA-Administrators
   };
   ```

4. Let authenticated users update their own entries:

   ```
   set reg-user owners {
       own-subtree
       subtree=<o ACME><ou staff>
       perms=modify
   };
   ```

5. Let PABX operators update all telephone numbers (assuming a role called pabx-operators has been defined):

   ```
   set admin-user pabx-operators {
       role=pabx-operators
       subtree=<o ACME>
       attrs=telephoneNumber
       perms=modify
   };
   ```

6. Let public users view (but not change) the name, e-mail address, and telephone number of staff:

   ```
   set public-user public-staff-info = {
       subtree = <o ACME><ou staff>
       attrs=commonName,eMailAddress,telephoneNumber
   };
   ```

7. Let public users view (but not change) anything in the *public* subtree:

   ```
   set public-user public-info = {
       subtree = <o ACME><ou public>
   };
   ```

8. Make passwords invisible to everyone except DSA administrators (this requires that only DSA administrators have the necessary access controls)

    ```
    set protected-items passwords {
        subtree=<o ACME>
        attrs = userPassword
    };
    ```

9. Make users administrators of their own passwords so they have update rights:

    ```
    set admin-user = {
        own-entry
        subtree=<o ACME><ou staff>
        attrs = userPassword
    };
    ```

How Role-Based Access Controls Work

If you have set up a role, then you can assign that role to one or more access control rules. All members of that role then inherit those access control rules. This works for both static and dynamic roles.

This lets you manage security for large numbers of users.

When role-based access controls are in use, the following occurs:

1. A user sets up a binding with a DSA.
2. The DSA searches all roles in the Role subtree, looking for the user's DN in the *member* and *uniqueMember* attributes of any entry with the *groupOfNames* or *groupofUniqueNames* object classes.
3. The DSA uses the names returned by this search as the roles for that user for the duration of this binding, and uses them in access control decisions.

Note: Before you can use role-based access controls, you must set up roles in the directory.

More information:

Set Up Groups and Roles (see page 217)

Role-Based Access Controls in a Router System

In a distributed DSA environment, a router DSA can be set to perform role lookups using its own credentials.

The roles DSA must contain an access control rule that gives the router DSA access to the role subtree. This means that when a user connects to a local DSA and the password is confirmed, the DSA looks up the user's roles without bypassing access controls.

More information:

Set Up Role-Based Access Controls in a Router DSA (see page 168)

Assign a Role to an Access Control Rule

If you assign a role to an access control rule, then the access control rule applies to all users in that role when they connect to the directory.

Because you assign a role to an access control by creating an access control rule, you need to restart the DSA for the assignment to take effect.

To assign a role to an access control rule

1. Set up a static or dynamic role in the directory (see page 217).
2. Enable access controls.
3. Create an access control rule that specifies that the role is a user of the access control rule.
4. Save the changes to the access file.
5. Restart the DSA.

Example: Configure Role-Based Access Control in a Router DSA

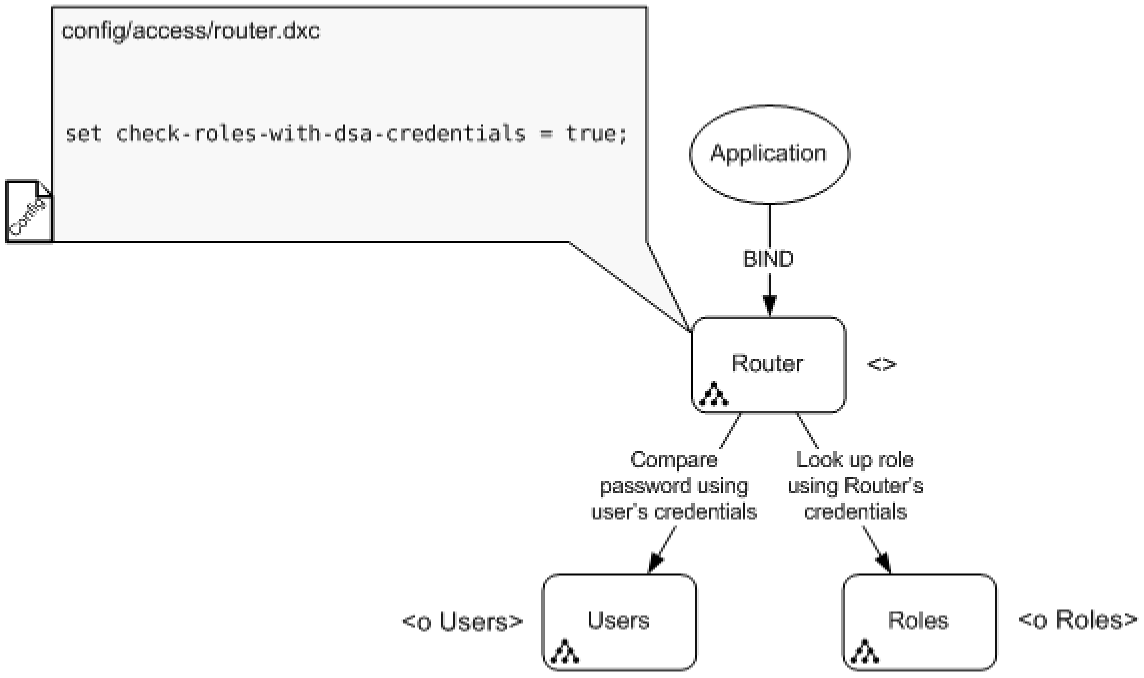
In this example, you have set up a distributed directory with the following three DSAs:

- A router DSA
- A DSA containing the entries of users
- A DSA containing the roles

Both the Router and Roles DSAs are configured with the following commands:

```
clear access;
set access-controls = true;
set role-subtree = <c AU><o Roles>;
set use-roles = true;

set reg-user = {
    user = <c AU><cn DXserver>
    subtree = <c AU><o Roles>
};
```


config/access/router.dxc
set check-roles-with-dsa-credentials = true;
Config
Application
BIND
Router
<>
Compare password using user's credentials
Look up role using Router's credentials
<o Users>
Users
Roles
<o Roles>

How Secure Proxies Work with Access Controls

The secure proxy feature lets an authorized user (or process) impersonate a user or role for whom it has responsibility, and bind to a DSA to elicit access control decisions, for example, a web server using a directory as an authorization engine.

The proxy binds to a DSA using certificate-based authentication. Those users permitted to proxy (impersonate someone else) are identified to a DSA as a list of distinguished names held in the *trust-sasl-proxy* list.

When using a proxy, impersonating usera can never obtain more power than they already have. In other words, in assuming the identity of another user, they may not obtain all the privileges of that user.

Once bound to a DSA, the proxy may change identity by changing the value of *dxProxyUser* in the entry *cn=roles* to the distinguished name of the new identity. This has the effect of evaluating the roles for the new identity. An exception is where the proxy also changes the value of *dxProxyRole* in cn=roles. In this case, roles are taken directly from the attribute value, not by accessing role entries in the directory.

The entry cn=roles is an operational entry, which means that it does not appear in the directory information tree, and it is never returned in any query results.

To permit the proxy to impersonate users not in the directory (external users), the proxy replaces the value of the *dxProxyUser* attribute in the operational entry with the distinguished name of the new identity, and at the same time replaces the value of the *dxProxyRole* attribute with the roles of the new identity. In this instance, the roles provided in the replace operation are used directly, if the following command is set:

```
set ssl-auth-bypass-entry-check = true;
```

Configure Access Controls

The following sections describe how to set up static access controls.

For more information on each command type and available parameters, see DXserver Commands.

Enable Access Controls

Use the *set access-control* command to turn access controls on. This command is a prerequisite for using access controls within CA Directory.

To enable access controls, use the following command:

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

Disable Access Controls

Use the *set access-control* command to disable access controls.

When you disable access controls, all users have complete access to the directory.

Use the following command to disable access controls:

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

Create Access Control Rules

Each access control rule is a command in a configuration file. When a DSA starts, it reads its configuration files, and applies the settings it finds.

Note: You can not enter control rules directly in the CA Directory console.

To set up access control rules

1. Enable access controls (see page 167).
2. Set up access control rules for one or more of the following access levels:

   Super users

   Super users have unrestricted read and update access to all parts of the DSA.

   To set access controls for super users, use the command *set super-user.*

   Administrative users

   Administrative users typically have read and update privileges over a specified directory subtree.

   To set access controls for administrative users, use the command *set admin-user.*

   Protected items

   Protected items take away privileges from registered users and public users.

   To set access controls for protected items, use the command *set protected-items.*

   Registered users

   Registered users typically have read access to a specified subtree.

   To set access controls for registered users, use the command *set reg-user.*

   Public users

   Public (anonymous) users typically have read-only access to a specified subtree.

   To set access controls for public users, use the command *set public-user.*

3. Save the access configuration file.
4. Restart the DSA.

Set Up Role-Based Access Controls in a Router DSA

To use role-based access controls in a distributed environment, you can configure a router DSA to compare passwords and look up roles using its own credentials. Do this in DXmanager, by setting the flag *Trust DSA_triggered operations*.

More information:

Role-Based Access Controls in a Router System (see page 164)

Display Access Controls

Use the *get access* command to view all current access control details.

To display access controls

1. Use Telnet to connect to the DSA that you want to check.
2. Display the access controls using the following command:

   ```
   get access;
   ```

Clear Access Controls

Use the *clear access* command to clear all existing access controls (or rules). The *clear access* command is commonly used when redesigning your access control policy. It ensures there are no existing access controls in place that may override your new ones.

To clear all access controls, use the following command:

```
clear access;
```

Check Whether Access Controls Are Enabled

Use the *get oper* command to view all current access control details.

To check whether access controls are enabled

1. Use Telnet to connect to the DSA that you want to check.
2. Display the operations settings using the following command:

   ```
   get oper;
   ```

   If the output from this command includes the following line, access controls *are* enabled:

   ```
   access-controls = TRUE
   ```

Chapter 9: Set Up Encryption

This section contains the following topics:

SSL

SSL is a protocol for providing secure communications over TCP/IP. With its successor Transport Layer Security (TLS), SSL has become the industry standard for confidentiality and integrity services.

For more information about SSL, see the SSL 3.0 Specification.

CA Directory supports SSL methods for encryption and authentication. These are known as SSL encryption and SSL authentication. SSL authentication always uses SSL encryption.

You can use SSL encryption to protect any link connections to or from CA Directory DSAs and web applications.

SSL supports X.509 certificates. You can manage these certificates by using DXcertgen, a certification authority, or other appropriate software.

SSL Transactions

An SSL transaction is between a server and a client. An example is when an LDAP client has a transaction with a DSA. The DSA is a server in the SSL transaction.

How an SSL Encryption Connection Is Established

In an SSL transaction, the following happens:

1. The client sends a handshake to the server.
2. The server sends its certificate to the client.

   The certificate includes the following information:

   - The DN of the DSA
   - The server's public key
   - The DN of the Certification Authority that issued the certificate
   - The time for which the certificate is valid

3. If the certificate is valid and the client trusts the Certification Authority, the client has now verified the server's identity, and the connection is made.

When to Use SSL Encryption

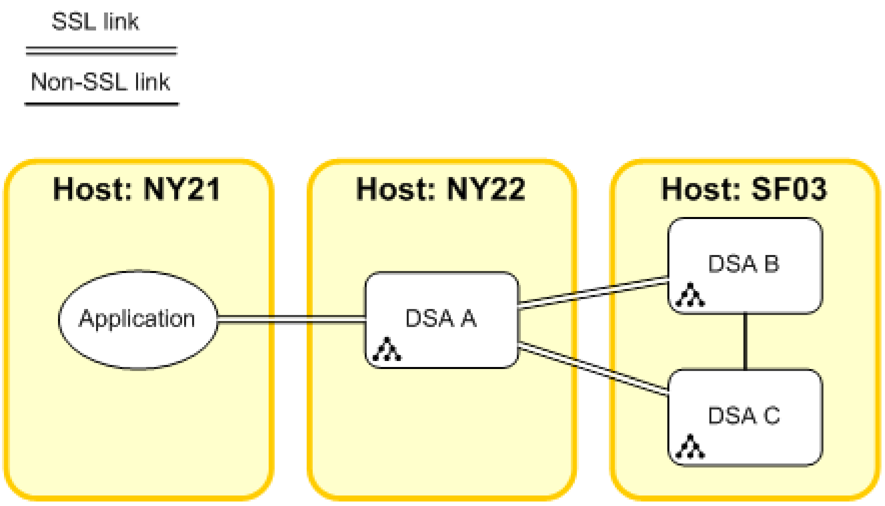
We recommend that you do not use SSL encryption between DSAs on the same computer, because the computer should already be secure, and SSL encryption is CPU-intensive.

However, you should use SSL to protect connections between clients and DSAs on different computers.

Example: Links between DSAs and an Application

The following diagram shows an application and directory backbone that are spread across three computers.

In this backbone, SSL encryption is used for all links between computers, but not for the link between the DSAs on SF03:



Certificates and Keys

To take part in an SSL session an SSL server needs to publish a certificate, and needs to hold the corresponding private key.

A certificate is a digital document that contains text encrypted with a private key. If you change the certificate without knowing the private key, you invalidate the certificate. If the certificate is valid, anybody can use the public key to decrypt the certificate information. If the public key does not work then somebody has corrupted the certificate and it cannot be trusted.

In CA Directory the certificate subject incorporates the DSA name.

You cannot use a certificate that is valid for one DSA on another DSA.

You can only create a valid certificate by using appropriate software, for example the CA Directory tool DXcertgen or OPENSSL.

All certificates used for CA Directory must be in X.509 PEM format.

SSL Processing

CA Directory handles DAP, LDAP, DSP and DISP protocols over an SSL connection. Each DSA will process SSL connections internally.

SSL processing is a based on OpenSSL. The OpenSSL library has been modified to limit the cipher suites to those that comply with U.S. export controls and the terms of the CA Directory encryption export license, and to include HSM support. Symmetric keys are limited to 256 bits or less, and asymmetric keys, used in key exchange, are limited to 2048 bits.

CA Directory supports the following protocols:

- SSL v2
- SSL v3
- Transport Layer Security (TLS)

How DXserver Uses Certificates

The DXserver uses the following types of certificates, which are stored in its configuration directory, DXHOME/config/ssld:

DSA certificates

There is one certificate for each DSA, and each one contains a private key.

Root CA certificates

These are one or more certificates of trusted Certification Authorities.

User certificates

These certificates are used by users (clients), for example, JXplorer.

DSA Certificates

To take part in SSL as a server, a DSA needs to have a certificate and the corresponding private key. CA Directory always stores the DSA certificate in a PEM file.

CA Directory can store the private key with the certificate, or you can specify that it stores the private key in a Hardware Security Module (HSM).

DXserver uses the DSA name to find the DSA's certificate.

If private key is stored in a PEM file, DXserver uses the DSA name to determine the name of the PEM file to access and use the certificate.

If the private key is stored in an HSM, then the PEM file holds a certificate but not the private key. DXserver uses the DSA name to determine the name of the PEM file and uses the HSM to access and apply the private key when this is required.

In either case the PEM file is named *dsaname*.pem (the DSA name is converted to lowercase).

How to Create a DSA Certificate

To create a certificate for a DSA, you need to provide the DSA name. The DSA must already exist.

Use this process to create a certificate for a DSA:

1. Use a certificate authority that you trust to create a certificate and the associated key pair.

   You can use the following tools:

   - DXcertgen
   - Certificate Authority software such as OpenSSL, which is available from the OpenSSL site.

2. Check the created certificate to ensure that the DSA name in the subject field of the certificate matches the DSA name.
3. Add the root certificate of the Certificate Authority that you used to create the DSA certificate to the end of the file trusted.pem.

How to Store DSA Keys in an HSM

You can store private keys in an HSM instead of a file. DXserver accesses the HSM using Public Key Cryptography Standard 11 (PKCS#11).

CA Directory is designed to support any HSM that supports PKCS#11. It has been tested on the Eracom “ProtectServer Orange External”.

To store keys in an HSM that has an onboard CA engine to create private keys and export the signed certificates, use the supporting tools from the HSM manufacturer as follows:

1. Generate the DSA key pairs in the HSM
2. Get the HSM to sign a certificate request. The subject in the certificate must be the DSA name.
3. Export the certificate in PEM format.
4. Name the PEM file to the DSA name, converted to lowercase, with the added extension .pem
5. Copy the PEM file to the ssld config directory.
6. Export the root CA certificate from the HSM and store it in the file trusted.pem.

When an SSL session occurs, DXserver uses the certificate subject and the HSM pin number to access the HSM.

About the DXcertgen Tool

CA Directory includes the DXcertgen tool for generating and working with certificates and their associated keys. You can use DXcertgen to do the following:

- Generate new certificates and key pairs for DSAs and users
- Report on currently configured certificates
- Import a third-party CA certificate
- List and delete CA certificates stored in a PEM format
- Create a certificate signing request for an external Certificate Authority

How DXcertgen Creates Certificates When There Is No Keystore

When you use DXcertgen to create certificates, and do not use a keystore, Dxcertgen does the following:

1. Creates a new key pair and root certificate, and stores these in memory.
2. Generates the user and DSA certificates, and signs them with the private key.

   For DSA certificates, the DSA name is used as the subject of the certificate.
3. Destroys the private key and appends the root certificate to the end of the following file:

   ```
   DXHOME/config/ssld/personalities/trusted.pem
   ```

Every time that DXcertgen creates a certificate, and does not use a keystore, it creates a new key pair and root certificate, and reissues all the user and DSA certificates based on the new key pair. This can be inconvenient, because you need to distribute the certificates to the client applications.

How DXcertgen Creates Certificates When There Is a Keystore

When you use DXcertgen to create certificates and specify that DXcertgen should use a keystore, the following happens:

1. If DXcertgen cannot find the root certificate or the key pair in the keystore, it creates a new root certificate and key pair, and stores these in the keystore.
2. When DXcertgen finds the root certificate and key pair in the keystore, it uses the root certificate and the private key to sign the certificates.

By default, DXcertgen looks for the keystore in the default CA Directory ssld config folder, which is as follows:

```
DXHOME/config/ssld/javakeystores
```

However, when you invoke DXcertgen, you can use options in the dxcertgen command to specify a different location.

Enable DXcertgen to Use a Keystore

Before DXcertgen can use a keystore you need to set the environment for it.

To Enable DXcertgen to Use a Keystore

1. Install the Java Runtime Environment (JRE)

   You need JRE because it contains the keytool utility that DXcertgen uses to manage the keystore.

   You can install JRE from the CA Directory installation CD.
2. Set the environment variable JAVA_HOME to the JRE path, for example:

   ```
   C:\Program Files\Java\jre1.6.0_16
   ```
3. Ensure that the environment variable PATH includes the Java bin folder, for example:

   ```
   C:\Program Files\Java\jre1.6.0_16\bin
   ```
4. To verify, run the following command:

   ```
   keytool
   ```

Note: The keytool utility is a Java utility and is documented on the java keytool web page on the Java web site http://java.sun.com.

Specify a Keystore for DXcertgen

CA Directory is shipped with a keystore for DXcertgen to use. You can specify this keystore, or another one, for DXcertgen to use to store the root certificate and key pair.

To Specify a Keystore for DXcertgen

1. Store the certificate and private key in a keystore

   You can use the keytool command to do this.

2. When you run DXcertgen, specify the path to the keystore and the root alias of the certificate you created.

Example: Using a Previously Created Root Certificate Stored in a Keystore

Assume you have stored a certificate and private key as follows:

- The keystore is DXHOME/config/ssld/javakeystores
- The access password to the keystore is keystorePassword
- The stored certificate has an alias myalias

The following command then causes DXcertgen to use that certificate to create certificates and private keys for all configured DSAs:

```
dxcertgen -a myalias -s DXHOME/config/ssld/javakeystores -S keystorePassword
```

Report on Certificates

DXcertgen can report on the status of the directory's certificates.

The report shows who the subject of the certificate is, the Certificate Authority who issued it, validity dates, and whether the certificate is currently valid.

To Report on Certificates

1. Enter the following command:

   ```
   dxcertgen report
   ```

2. Check that the status of all the certificates is *valid*.

   If the status of any certificate is *invalid*, you should generate a new certificate.

Use DXcertgen to Request and Use a Third-party Certificate for a DSA Certificate

By default, Dxcertgen creates a certificate and associated private key for each DSA and user. However, you can use Dxcertgen to request another Certificate Authority (CA) to create a certificate. You can then use DXcertgen to merge the DXcertgen private key with the CA-created certificate to create a DSA certificate and associated private key.

To Request and Use a Third-party Certificate for a DSA Certificate

1. Create a Certificate Signing Request (CSR). Use the following command:

   ```
   dxcertgen -D dsaname certreq
   ```

   Dxcertgen creates a private key to create the CSR.

   The command stores the CSR in DXHOME/config/ssld/*dsaname*.csr, and stores the private key in DXHOME/config/ssld/*dsaname*.key.

   Keep the private key, but send the CSR to the CA.

   Note: For the CA to sign the CSR, they might need additional information such as validity timeframes.

2. When the CA responds with a certificate, use the following command to merge this certificate with the private key that was used to create the CSR:

   ```
   dxcertgen -D dsaname -n ca_response_certfile certmerge
   ```

   DXcertgen stores the certificate and private key in *DXHOME/config/ssld/personalities/dsaname*.pem.

3. Add the certificate to the file trusted.pem. CA Directory trusts the certificate that you have created. You can use the following command to do this:

   ```
   dxcertgen -n ca_response_certfile importca
   ```

   Note: Use the file trusted.pem to contain root certificates only.

Delete a CA Certificate

The CA certificates are stored in the file trusted.pem.

Where there are multiple certificates in a file, they are numbered in the order they are found. When you add and remove certificates in trusted.pem, the numbering changes. Ensure that you list the certificates immediately before you delete a certificate.

To delete a root certificate

1. List the certificates that are stored in the file trusted.pem, by using the following command:

   ```
   dxcertgen listca
   ```

2. Find the certificate that you want to delete, and note the number of this certificate.
3. Delete the certificate, using the following command:

   ```
   dxcertgen -r certnumber removeca
   ```

How to Encrypt Communication between CA Directory Components

You should use SSL to protect connections between components on different computers.

The following diagram shows these secure connections:

1. Between the browser and both DXmanager and JXweb

To make this link secure, encrypt communications between your browser and DXwebserver (see page 185) or JXweb.

2. Between DXmanager and the directory

The link between DXmanager and the directory is always secure.

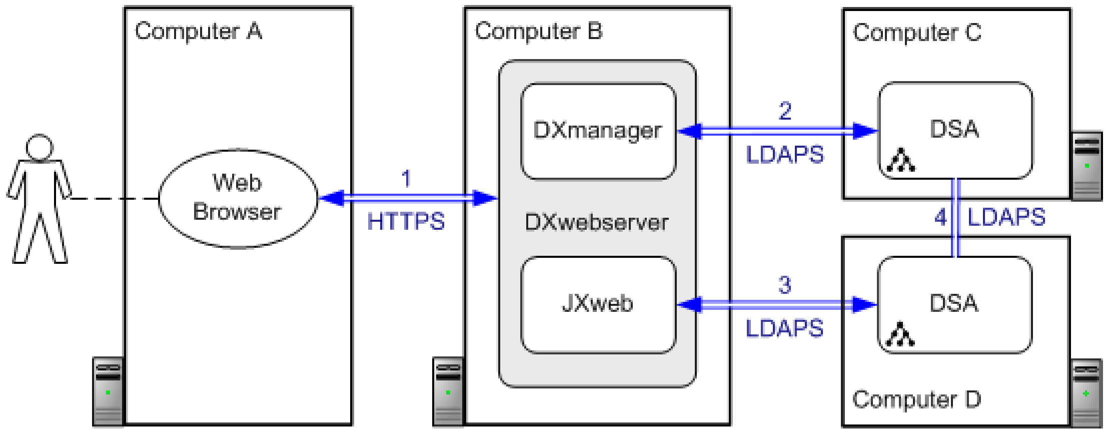
DXmanager uses LDAPS to communicate with the directory.

3. Between JXweb and the directory

To make this link secure, encrypt communications between JXweb and the directory (see page 186).

4. Between DSAs on different computers

To make this link secure, encrypt communications between DSAs.



How to Encrypt Communication to DSAs

To encrypt communication to DSAs, you need a root certificate, a DSA certificate.

1. Set up the certificates.
2. Ensure that each DXserver knows the following:
   - The port on which it listens for requests
   - The location of the DSA and user certificates, and public and private keys
   - The location of the root certificate

   Note: If you are using an HSM you must also provide the HSM specific information.

Encrypt LDAP Bindings

You can force SSL encryption over LDAP links for both anonymous and authenticated bindings.

To force SSL encryption on anonymous bindings, include the following command in the settings configuration file of the DSA:

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

When this setting is on, if a user tries to create an anonymous binding without SSL, the DSA disallows it and returns an "Inappropriate authentication" error.

To force SSL encryption on authenticated bindings, include the following command in the settings configuration file of the DSA:

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

When this setting is on, if a user tries to create an authenticated binding without SSL, the DSA disallows it and returns an "Inappropriate authentication" error.

The *set force-encrypt-auth* setting does not prevent the credentials from being sent unencrypted over the network. However it refuses any unencrypted binding request.

Configure DSA to Act as an LDAP Client with SSL Encryption

You can set up SSL encryption between third-party LDAP servers and your CA Directory backbone.

CA Directory DSAs operate as SSL clients when they act as LDAP clients to communicate with LDAP servers. This means that the LDAP servers do not need copies of the CA Directory DSA's root or DSA certificates.

You can secure the connection using SSL encryption by using the following steps:

1. Ensure that you have access to a Certificate Authority.
2. Using your Certificate Authority, produce server certificates for both the LDAP server and the DSA, and sign them with the Certificate Authority's root certificate.
3. Configure CA Directory and the LDAP server to trust the Certificate Authority by importing the root certificate.
4. Configure CA Directory and the LDAP server to use the server certificate signed by the Certificate Authority for SSL operations.
5. Configure CA Directory to connect to the LDAP server by using DXmanager.
6. Test that everything is working correctly, as follows:
   a. Start the LDAP server.
   b. Start the DSA.
   c. Verify that SSL is being used using the following command on a DSA console:

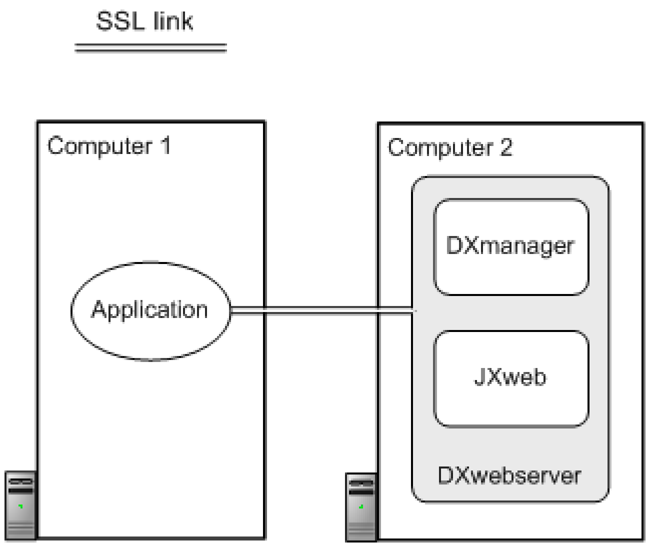
      ```
      trace x500;
      ```

      SSL operations are now prefixed with **(SSL)**.

How to Encrypt Communications Between Your Browser and DXmanager or JXweb

You should encrypt communications between your browser and DXmanager or JXweb, because these communications might include sensitive information, including port numbers, DSA passwords, and DSA console passwords.

To secure these communications, use SSL between your browser and DXwebserver, as shown in the following diagram:



1. Create a keystore for DXmanager (see page 186).
2. Create a Certificate Signing Request (CSR) using the command dxcertgen certreq
3. Send the CSA to a trusted certificate authority to verify and sign.
4. (Optional - Windows only) Update any shortcuts to DXwebserver with the SSL connection details.

Create a Keystore for DXmanager

If you want to encrypt data between the web browser and DXmanager. you need a keystore for DXmanager.

To create a keystore for DXmanager

1. Ensure you have the Java *bin* directory set in your path.
2. Open a command prompt.
3. Run the *keytool* command from your home directory.

   For example, use the following command:

   ```
   keytool -keystore dxmanager.iks -storepass changeit -keypass changeit -alias
   dxmanager -genkey -dname "cn=dxmanager,ou=CA Directory, o=CA, c=AU" -alias
   dxmanager -validity 100 -keysize 512
   ```

   You have created a keystore file.

For information about more command options, visit the keytool page on the Java site.

How to Encrypt Communications between JXweb and the Directory

JXweb lets you use a secure link to connect to a directory. You set this each time you connect.

To encrypt communications between JXweb and the directory

1. Configure DXserver for SSL.
2. Start JXweb.
3. In the Connect dialog, select the Use SSL check box.

   When you click Connect, JXweb connects to the directory using an SSL-encrypted link.

Configure a DSA to Use SSL

DXserver handles SSL and TLS authentication, encryption, and decryption.

To configure a DSA to use SSL

1. Stop the DSA.
2. Add the following command to the settings file sourced by the DSA:

   ```
   set ssl = {cert-dir = CertificationDirectory ca-file = CertificateAuthorityFile};
   ```
3. Start the DSA.

You can find the default SSL settings in the file $DXHOME/config/ssld/default.dxc. If you do not set a cipher, the system uses the default ciphers. You can check the available ciphers using the console command *get ciphers.*

For a list of ciphers currently supported, see the section Encryption Formats for SSL.

For more information about ciphers, see the OpenSSL site.

Configure the DXtools to Use SSL

By default, you cannot connect to a directory using SSL with the following tools:

- DXsearch
- DXmodify
- DXrename
- DXdelete

None of these tools can see the DSA's trusted.pem file and therefore are unable to trust the self-signed certificate.

To configure the DXtools to use SSL

1. Create a new text file with the following name and location:

   ```
   DXHOME/config/ssld/dxldap.conf
   ```

2. Open the file in a text editor, and add the following text:

   ```
   TLS_CACERT  c:\program files \CA\Directory\dxserver\config\ssld\trusted.pem
   TLS_REQCERT never
   ```

3. Save and close the file.
4. Set an environment variable named LDAPCONF that points to the file you have just created.

   You should now be able to connect to the directory through SSL using the tools listed.

Chapter 10: Set Up Views

This section contains the following topics:

About Views

A view lets you make a series of searches on a directory, and have the results returned together. Each phase of the search can use the results from previous phases as its inputs.

A view provides a virtual abstraction of directory data. This enables LDAP applications to view the data in a way that matches the needs of the application, similar to the way that database views are used in RDBMS systems. You define a view to the DSA process and can then invoke it multiple times.

Sometimes you need the results from one search to use as input to another one, and you may also need to include the results of multiple searches to get the result you need. A view helps in such situations because it gives you a way to define a single complex search that combines a number of ordinary LDAP searches. The searches can be performed across multiple DSAs, and a search can use the results of earlier searches in the view.

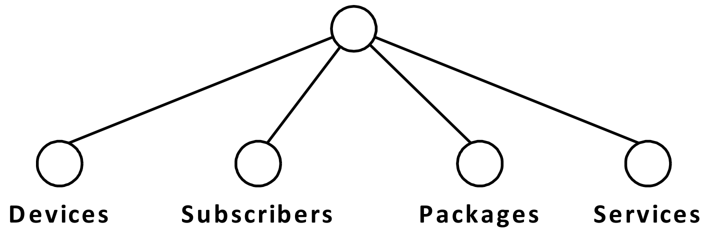
A view is a complex command, which can be stored in a configuration file. When you need to use a view you search the directory, invoking the view in the search command. The directory carries out the search, including the phases defined in the view. When the searches are all complete, it return the results in a single response.

Some advantages of views are as follows:

- Views allow simpler applications, and let you keep the same application even if the underlying data changes.
- Views let you normalize directory information, which reduces DIT size, and still lets you reference the data with a single, view-based, search.
- Views increase performance because the DSA keeps intermediate results for reuse in a view, rather than sending them back to the user.

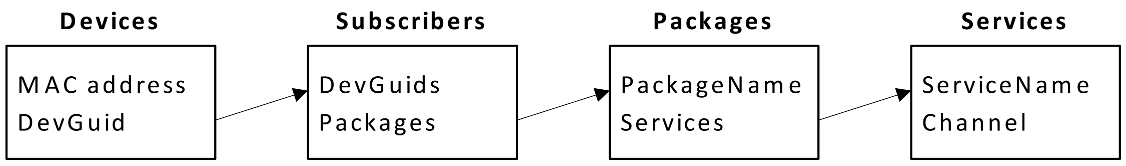
Example: A View Used by a Pay TV Company

In this example, the directory for a pay TV company stores information about devices, subscribers, the packages they subscribe to, and the services included in each package in separate places in the DIT:



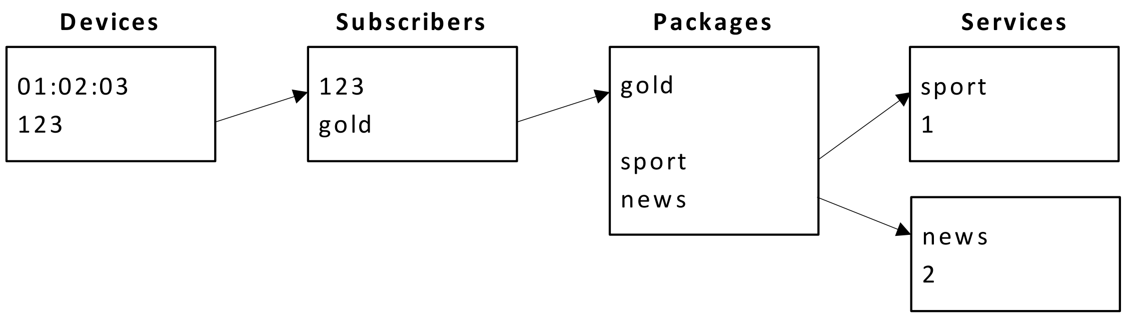
The structure of this directory makes it simplefor staff to navigate, but it is not easy to find information about the services associated with a particular device.

The following diagram shows the series of searches required to find out the channels being served to a particular device:



You can define a view so that a search for a device's MAC address can return the channels available on the device.

The following diagram shows the results of such a search:



Compare Views to Stored Procedures in a Database

Views allow relational concepts (well understood in databases) to be possible in directories, which have always been associated with fixed schemas and hierarchies.

This table compares views in CA Directory with stored procedures in databases:

Stored Procedure in a Database	View in CA Directory
A database application queries a stored procedure using its name	A directory application queries a view using its name (DN)
The SQL query to the stored object may have parameters (via the Where clause)	The LDAP query to the view object may have parameters (via the filter)
The stored procedure executes all instructions which may be conditional and/or branching etc.	The view template gets executed, invoking each phase which may be conditional and/or branching etc.
The stored procedure returns all/selected/pruned results	The View returns all/selected/pruned results

Limitations of Views

Views have the following limitations:

- Access controls - All phases of the view searches are performed with the credentials of the user invoking the view.
- Views may require specialized applications for performing view searches and interpreting results
- Views require provisioning applications to be aware of the relationship between objects
- Unexpected results may occur when the linking attributes are not unique for a given subtree (this can be enforced by using 'unique-attrs')
- Mapping - the attributes used to link objects don't require the same type but need to be the same (normalized) value

Ways to Use Views

This section goes through a number of real-world examples and how views can be harnessed to meet these requirements.

Using Views for Data Normalization (Class of Service)

Views can be useful if your directory contains highly repetitive information. For example, each user entry contains office location and contact details which are identical for many users. This uses a lot of space for caching and backups and is complex to update.

To solve this, use views in the following way:

1. Create a separate subtree to hold office information including a unique office identifier in each entry.
2. Replace the repetitive office information with the related office identifier in each user entry.
3. Create a view so that an entry being returned will also perform a lookup of the office details and include them in the user entry.

Using Views for User Preferences

If your directory one-to-many subordinate entries that each contains various preference information, you could use a view to perform a single search returning the entry you are interested in, and all its subordinates.

To do this, create a view that retrieves a target entry.

The result of this search (base-object of entry returned) can then be used to perform a one-level/subtree search to return the entry's subordinates.

Using Views to Restrict Access

A view can remove values before it returns the result. This can be useful for restricting access.

For example, a pay TV customer that has subscribed to a package that includes adult content, wishes to have adult content disallowed as they have small children.

A view can be created to retrieve the services for a particular package a user has subscribes to.

The next phase of the view can retrieve restricted items that can then be prune values/objects retrieved above.

Using Views to Overlay Another Directory

An overlay directory lets you augment entries in a another directory with extra attributes. This lets you use CA Directory to extend the schema of an existing less-flexible directory.

Views can be used to achieve this by storing entries in another directory instance that share unique values with the other directory. A view can be created to retrieve the two linked entries and merge the result into a single entry.

How Views Work

Example: Cell Phone Service Provider

Consider a cell phone provider. To find the phone's SIM card number in the provider's directory structure, you first find the customer's cell phone number and then use that to find the SIM card number. The following table shows how to do the complete task using a view.

Define View Command	Annotation

`set view "SIM View" = {`
`description = "Display Sim Number given a name"`
	This line defines a view and gives it a name. The name and description are displayed from the *get view* command.

`entry = <o ACME><ou Views><cn "SIM">`	This line defines the base object DN for this view. The user specifies this virtual entry to invoke the view, so to invoke this view with the ldapsearch command, the command line must include the following option: `-b "cn=SIM,ou=Views,o=ACME"` In this organization (ACME), the administrator has included an organizational unit called Views solely to ensure that all the view DNs are unique.

`(phase = 1`
` subtree = "ou=Customers,o=ACME"`
` filter = "(cn=$cn)"`
` ),`
	This is the first phase of the view, so it must be labeled 1.
	When the view is invoked, the phase searches the subtree starting at the following DN:
	`"ou=Customers,o=ACME"`
	The use of the term *$cn* in the phase means that users must specify the value of *cn* when they invoke the view. For example, the search could include the filter:
	`(cn="John Smith")`
	The DSA uses this value to replace *$cn* term. Therefore, phase 1 performs a search with the following filter:
	`(cn="John Smith")`
	Because the phase does not include any other options, the phase returns all the information in the subtree.

Define View Command	Annotation

`(phase = 2`
` subtree =`
`"ou=cellphones,ou=Accounts,o=ACME"`
` filter = "(account=$1:cellphone)"`
` eis = simCardNumber`
` options = result-required`
` )`
`};`
	This is the second phase of the view, so it must be labeled 2.
	This phase searches the following subtree:
	`"ou=cellphones,ou=Accounts,o=ACME"`
	The DSA replaces *$1:cellphone* with the value of the *cellphone* attribute returned by phase 1, for example, 01001001001
	The *eis* option means that the only attribute value that the phase returns is *simCardnumber*.
	If the customer has two phones, then the filter is an OR of the phone numbers returned, as in the following example:
	`(|(account=01001001001)(account=01001001002)`
	Two simCardNumber values are returned in this case.
	Because this phase specifies *result-required*, the DSA checks that the link between the attributes in each subtree is valid. If the link is not valid, then it raises an alarm.

Given the view defined in the preceding table, you could use the following search to find a customer's SIM card numbers:

```
ldapsearch -h hostname:3000 -s subtree -b "cn=SIM,ou=Views,o=ACME" "(cn=John Smith)"
simCardNumber
```

How the DSA Processes Phases in a View

A view contains one or more phases, which are numbered. When the view is invoked, the DSA goes through each phase in the view, in order, and runs the phase's search. The power of views arises because a phase can use, as input to its search, results from a previous phase in the view. The DSA runs a phase as soon as the phase has all the input it requires.

You can specify that the DSA should use the output of one phase as input in another phase. You do this using the views parameters, which you include in the view definition in place of attribute values. The DSA replaces parameters with values as soon as the results are available, either from the values given in the invoking search command, or from the result of a phase.

The DSA runs a phase as soon as all the parameters it needs have been replaced by values, and it runs phases in parallel if possible.

The order of the phases is important because the DSA runs each phase only once per view invocation, and the DSA can only replace a phase's parameters with the results from earlier (lower numbered) phases. That is, a phase's parameters cannot refer to later phases.

How the DSA Compares Parameters in Views

When the DSA replaces a parameter with a value, it transforms the value to a canonical form that is based on the matching rules that are specified for the attribute. Matching rules define how to compare two strings, to test if they are equal, or if one is greater than or less than the other. RFC 4517 defines these matching rules.

All attribute values have associated matching rules. One commonly used rule is the *caseIgnoreMatch* rule, which says that you can ignore letter case when comparing strings for equality.

When a views parameter is compared to a phase result and the matching rules are not identical, the transformation that the DSA has performed can produce unexpected results. For example, spaces are not important for telephone numbers, so if an attribute is defined with *telephoneNumberMatching* matching rules, the following entries are equivalent:

```
+61 3 1300-1001
+61313001001
```

However, under *caseIgnoreMatch* rules these two strings do not match.

Similarly, you can invoke a view with the following search command:

```
ldapsearch... "ph="+613 1300-1001"
```

In this case, if *ph* is defined with telephone number syntax, the DSA stores *$ph* as follows:

```
+61313001001
```

If an account number is stored as a *numericString*, then the DSA transforms the following result:

```
ac="+613 1300-1001"
```

That result is converted to the following numeric string:

```
61313001001
```

Because comparisons are performed after the strings are converted, the following comparison fails:

```
($ac=$ph)
```

How a Phase Uses Multiple Results from an Earlier Phase

A phase can return multiple entries, and can also return an attribute with multiple values. How the DSA handles multiple results in later phases depends on where the later phase specifies the previous phase's result as defined in the *set view* command. In a phase definition of the *set view* command, the following options can specify the results of previous phases:

Subtree DN

A phase always has a subtree specification, which is the DN that defines the root of phase search. If this DN is defined by the result of a previous phase, then the DSA runs a separate phase search for each different result. For example, if phase one returns four DNs, and phase two specifies its subtree DN as the result of phase one, then the DSA runs a phase two search four times, once with each different value of DN.

Filter

If a phase specifies a filter as part of its search, then the DSA ORs all the specified values of all the returned entries. For example, consider a view in which phase one returns two entries with attribute Attr, the first of which has two values Val1 and Val2, and the second of which has value Val3. A filter in a subsequent phase can use the returned values of Attr from phase one by including the fragment:

```
filter = $1:attr
```

DSA replaces this with the following:

```
filter = (|(attr=val1)(attr=val2)(attr=val3))
```

eis

Every return value is used. This is equivalent to specifying multiple attribute names in the eis option.

allowed-attr, allowed-target, prune-attr, prune-target

All combinations of every value are used. For example if allowed-attr has three attribute names and allowed-target has two attribute names, then both attributes in allowed-target have three new values.

How the EIS Works

In a search filter, the *entry information selection* (EIS) is the attributes that are to be returned in the search results.

If an EIS is passed in on the view search then all phase searches will include this and include any linking attributes not present. This overrides the EIS specified for each phase.

If no EIS is passed in on the view search then the phase search will include any phase EIS attributes plus any linking attributes not present.

If a phase search returns multiple entries with the same linking attribute or a linking attribute with multiple values, then the searches will branch off. This occurs when the relationship between objects is one-to-many or many-to-many. When this occurs the next phase search's filter will contain an OR of the filter defined. For example, 2 entries are returned and 'attr' is the linking attribute. Entry 1 has attr = 1, attr = 2 and entry 2 has attr = 3. If the next phase search filter is (attr=$attr) then the search performed will have filter (|(attr=1)(attr=2)(attr=3)).

You can modify EIS in the following ways:

- Delimiting - '$3:restriction[|:1]' would retrieve field one from the value of restriction returned by the phase 3 search delimited by '|'. For example, the value "service|01012007|didn't pay bill" would translate to "service".
- Truncation - '3:restriction[5]' would retrieve the first 5 characters from the value of restriction.

In a view search, if the EIS is *dxEntryCount* then the result is the number of entries that on the view entry's base-object.

How to Use Views

You invoke a view by providing the view's DN as the base object in an LDAP request to the DSA. Search requests are restricted in scope to the base object and filtered subtree searches. Bind, compare, and modify requests are restricted to using only attributes contained in the entry defined in phase 1.

Design a View

Before views are configured the implementer needs to be aware of what objects are going to be stored and how they are related (data models). At this point use cases can be developed to solve particular application requirements.

Each object requires a separate search (phase). When each phase completes, then subsequent phases can be performed that rely on values returned. In the above example the phase search 1 will retrieve the 'DevGuid'. When this complete the phase 2 search can retrieve Packages using the 'DevGuid' returned, and so on.

Set Up a View

You define a view to the DSA process by using the *set view* command.

You might find it useful to create a file settings/views.dxc to contain your view definitions. This file should include a *clear view* command before the *set view* commands. See set view Command.

A view command has a header and a body. The header identifies the view and contains the view name, DN, and post-processing options. The view body consists of a list of searches, called phases. Each phase has a DN and a filter (and some ancillary information).

Example set view Command

This is the set view command used to create the view discussed in Example: A View Used by a Pay TV Company (see page 190):

```
clear view;

set view "Channel list" = {
   description = "Display list of channels for a given MACAddress"
   entry = <c AU><o Views><cn "Channel list">
   ( phase   = 1
     subtree = "o=Devices,c=AU"
     filter  = "(MACAddress=$MACAddress)"
     options = ignore-from-result ),
   ( phase   = 2
     subtree = "o=Subscribers,c=AU"
     filter  = "(DevGuids=$1:DevGuid)"
     eis     = Package
     options = result-required, ignore-from-result ),
   ( phase   = 3
     subtree = "o=Packages,c=AU"
     filter  = "(PackageName=$2:Package)" ),

   ( phase   = 4
     subtree = "o=Services,c=AU"
     filter  = "(ServiceName=$3:Services)"
     eis     = Channel )
};
```

Add a Prefix or Suffix to a Search Result

You can add a constant prefix or suffix to search results in a view.

For example, you could add a prefix to search results from a particular source, to help you distinguish these results from results from other sources.

You can add prefixes and suffixes to attributes with string syntaxes only (such as *caseIgnore* and *caseExact*).

To add a prefix or suffix to a value

1. Create a *set view* command.
2. In the *eis* section of one of the phases in the *set view* command, add a prefix or suffix as follows:

   ```
   eis = [prefix]attributeName[suffix]
   ```
3. For example, you could use this eis to add the prefix SAP to all results from this phase:

   ```
   eis = “SAP[$2:userId]” as uid
   ```

Example: Add the prefix AD to search results

In this example, the

```
eis = "Description=[$description]", "AD.[$initials], ">>>[$givenName]<<<" as
employeeType
```

How to Improve the Performance of a View

Searches phases that have all required information to proceed will be performed in parallel.

If 'ignore-from-result' specified then the EIS for this search only includes the linking attributes.

If 'prune-from-result' specified then the search will only bring back entries (no attributes)

Remove View Definitions

Before you define a view, you should remove any existing view definition that has the same name. However, there is no command to delete just one view. Instead, use this command, which removes all views. Normally, you put this command in the configuration file before any *set view* commands.

To remove view definitions

Issue the following command to the DSA:

```
clear view;
```

All view definitions in the DSA are removed.

How to Invoke a View

A view can be invoked by performing operations on the view entry or below.

You invoke the view by requesting a search that includes the DN of the view.

The invoking search request also specifies a filter and the scope. Both of these are passed to the view.

The DSA uses the filter only to supply values to parameters in the phases of the search.

The scope can be base-object search or a subtree search. The view does not directly use this scope specification, because the scope of each each of searches (phase) in a view is separately defined in the view definition. Instead, the view uses the scope to determine what filter to use in its first phase of the view.

Subtree Searches

When the scope is a subtree search, the filter is not used directly and the DSA uses the filter in the search request to pass parameters to phase one of the view.

Example: Invoking a View with a Subtree Search Command

In Pay TV example, the view would be invoked by the following command;

```
ldapsearch -h host -p 30000 -Lb "cn=Channel list,o=Views,c=AU" -s subtree
MACAddress=01:02:03 Channel
```

This search returns the following result if the 'collapse-result' option is enabled:

```
dn: MACAddress=01:02:03,cn=Channel list,o=Views,c=AU
Channel: 1
Channel: 2
```

Example: Invoking another View with a Subtree Search Command

```
ldapsearch -h host:30000 -b "cn=SIM,ou=Views,o=ACME" -s subtree (cn=John Smith)
SIMCardNumber
```

This invokes the Channel List view. Phase one of the view should have a filter that includes the term *$cn*. The DSA replaces any occurrence of *$cn* in the view with John Smith, and then runs phase one.

Base Object Searches

For a base-object search a filter containing (objectClass=*) can be used to read the entry and resolved the view. The filter in the phase 1 search is ignored for base-object searches, and the DSA uses the search filter you specify instead.

Base-object searches can be used to perform simple queries like (Channel=1) or (!(Channel=2)) which are applied after the view has been resolved.

Example: Base-object search

In the example above the view would be invoked by the following

```
ldapsearch -h host -p 30000 -Lb "MACAddress=01:02:03,cn=Channel list,o=Views,c=AU"
-s base (objectClass=*) Channel
```

This search returns the following result if the 'collapse-result' option is enabled:

```
dn: MACAddress=01:02:03,cn=Channel list,o=Views,c=AU
Channel: 1
Channel: 2
```

Example: Invocation of a View with a Base-Object Search Command

```
ldapsearch -h host:30000 -b "cn=SIM,ou=Views,o=ACME" -s baseobject (baseObject=*)
SIMCArdNumber
```

This invokes the Channel List view. The phase one filter is (baseObject=*).

Binds

Many applications perform a subtree search for a specific user. The user is then authenticated via a bind using the distinguished name of the entry returned.

Example: Bind request

If a subtree search of a view entry "cn=SMUsers,o=Views,c=AU" returns the view subordinate entry "guid=1234,cn=SMUsers,o=Views,c=AU", a bind for this user can be performed. The directory processes the bind in this manner::

1. The bind request is converted to a password compare of entry "guid=1234,ou=grp0,cn=SMUsers,o=Views,c=AU".
2. The phase 1 subtree is checked and found to be "ou=users,o=acme,c=AU".
3. The password compare is remapped to the phase 1 subtree, producing "guid=1234,ou=grp0,ou=users,o=acme,c=AU".
4. The password compare is applied to this entry and if the userPassword matches, the authentication is successful.

Modifies

When a user is authenticated by the view some applications update lastLogin information and for that user. When a modify request is received for a view subordinate entry the following processing will take place:

- The phase 1 subtree is checked "ou=users,o=acme,c=AU"
- The modify is remapped to the phase 1 subtree producing "guid=1234,ou=grp0,ou=users,o=acme,c=AU"
- The modify is applied to this entry

Note: Updates are only applied to the entry returned by the phase 1 search.

Compares

When a compare request is received for a view subordinate entry the following processing will take place:

- The phase 1 subtree is checked "ou=users,o=acme,c=AU"
- The compare is remapped to the phase 1 subtree producing "guid=1234,ou=grp0,ou=users,o=acme,c=AU"
- The compare is performed against this entry

Note: Compares are only applied to the entry returned by the phase 1 search.

Chapter 11: Set Up Groups and Roles

This section contains the following topics:

Groups and Roles

CA Directory lets you set up groups in a directory. You can use groups to allow applications to customize the user experience based on which groups they are a member of.

If you set up static or dynamic groups, you can also use these groups as roles. The directory can use role membership to determine a user's access permissions and configuration items within the directory itself. For more information, see Role-Based Configuration (see page 226).

Types of Groups and Roles

There are three kinds of groups in CA Directory. Static and dynamic groups are very similar, and we recommend that you use one of these rather than access control groups.

Static groups and roles

A *static group* is an entry in the directory with a *member* attribute, which stores a list of the DNs of the entries that are members of this group.

If you set up static groups, you can then use *static roles*, to which you can assign access controls.

Dynamic groups and roles

A *dynamic group* is an entry in the directory with its membership defined by an LDAP filter. All entries that satisfy this filter are members of the dynamic group.

If you set up dynamic groups, you can set up *dynamic roles* to which you can assign access controls.

Access control groups

Access control groups are defined in a configuration file. Each group includes a list of member DNs. You can assign access controls to each group.

Because the groups are in the DSA configuration, you must restart the DSA to add or delete groups, and to add or remove members from the group.

If you already use access control groups, you can keep using them. However, we recommend that for new groups, you use static or dynamic groups instead, because they are more flexible and powerful, for the following reasons:

- **Access control groups are less flexible**—To add, delete, or change members in a static or dynamic group, you need to edit only the group's entry in the directory. You do not need to restart the DSA as you do with changes to access control groups.
- **Access control groups are less powerful**—Applications can use static and dynamic groups because they are directory entries. You can also extend these groups to form roles, which the directory itself can use for access controls and role-specific configuration.

Comparison of Static and Dynamic Groups

The following table compares static and dynamic groups:

	Static Groups & Roles	**Dynamic Groups & Roles**
Reasons to use	■ The group membership is relatively stable.	■ The group membership changes frequently ■ Groups have large numbers of members
Advantages	■ Allows use of static roles	■ Allows use of dynamic roles ■ Easy to maintain (no need to add or delete members from groups) ■ No restriction on the number of members
Disadvantages	■ Hard to maintain (need to manually add and delete members from groups) ■ Update time increases as the number of group members increases ■ Data size may increase if the number of groups is very high and entries are included in many groups (due to members being listed many times)	■ Performance may decrease if the number of groups is very high (due to high number of group evaluations) ■ Applications cannot query group membership directly (instead, applications can read the group entry, and then search the members)

Static Groups and Roles

You can group users and other entries in the directory using a static group entry.

A static group entry contains a list of the group members. After you set up a static group, applications that query the directory can check if an entry is in a group, and use that information to set permissions for that entry.

If you want to set permissions and other configuration on an entry's access to the directory itself, you can extend the group to work as a role. You can then set permissions and limits for the role, and these apply to all of the role's members.

Static Groups

A static group is an entry in the directory that stores a list of the members of the group.

A static group entry usually has one of the object classes listed in the following table:

Object Class	Attribute Containing Members
groupOfNames	*member*
groupOfUniqueNames	*uniqueMember*

If a user is removed from the directory, then the user must also be removed from any static groups that the user belonged to. To do this, you remove the user's member DN from each group that the user was a member of. This means that static groups are useful for directories in which the group memberships do not change frequently.

If a static group contains a large number of members, it can take a long time to update the entry. Dynamic groups (see page 217) do not have this disadvantage.

How Static Groups Work

Static groups are entries in the directory that contain lists of the DNs of other entries.

An application can query the directory to check if a DN is contained in a particular group. The application can then use that information to grant or deny access to the user, or other changes.

Example: Static Groups for Application Administrators

This example describes how Company A, a large hotel management company, uses static groups in its staff directory.

Company A has hundreds of thousands of employees, all of whom are listed in the staff directory. The subtrees in this directory are divided by geography, and then by business unit.

The directory is used by many different applications, and each application needs to know which users should be allowed to administer that application. For example, the Payroll application gives extra privileges to users who are in the Payroll Administrators group.

When an employee logs in to the Payroll application, the following happens:

1. The user logs in to the Payroll application.
2. The application connects to the directory and searches for the user.
3. The application takes the DN of the user entry and searches the *groups* subtree for any groups in which this user is a member.

   In this example, the user is *not* in the Payroll Administrators group.
4. The application gives users the access only allowed to general employees. This might include permission to update their own address and view their salary, tax, and leave details.
5. If this user *were* in the Payroll Administrators group, the application would have given greater privileges, such as the ability to view and change the details of other employees.

How Static Roles Work

Static roles are an extension of static groups. Static roles let you control access to the directory itself, using groups in the directory.

To use static roles, you first need to set up static groups and then configure the roles.

When a user logs in to a directory with static roles, the following happens:

1. The user logs in to the directory.
2. The DSA authenticates the user.
3. The DSA searches all of the groups in the specified subtree for the user's DN.
4. If it finds any group that contains the user's DN in the *member* or *uniqueMember* attributes, it uses those groups as the user's roles for this connection.

   These roles are used in decisions about access controls and other role-specific configuration items.

If an access control or other configuration item is set for the whole DSA and for a user's role, the more generous setting applies to that user.

If you change the value of a role-based configuration item, the new value does not apply to users that are currently logged in. After they have logged out and back in again, the new settings apply.

Active Directory memberOf Attribute

CA Directory emulates the ability of Active Directory to auto populate the memberOf attribute when it returns or looks up user entries. The memberOf attribute contains all the group distinguished names (DNs) of which the entry is a member.

CA Directory updates the memberOf attribute of an entry every time the entry DN is included or removed from a group.

Example: Show memberOf information for a group

The following example returns the groups that *jsmith01* is a member of. If an entry is in the administrator and backup operator groups, then returning the entry *cn=jsmith01* includes the group that *jsmith01* is a member of:

```
dn: cn=Administrators,ou=Groups,o=CA,c=AU
member: cn=jsmith01,ou=Users,o=CA,c=AU
member: ...

dn: cn=Backup Operators,ou=Groups,o=CA,c=AU
member:  cn=jsmith01,ou=Users,o=CA,c=AU
member: ...

dn: cn=jsmith01,ou=Users,o=CA,c=AU
memberOf: cn=Administrators,ou=Groups,o=CA,c=AU
memberOf: cn=Backup Operators,ou=Groups,o=CA,c=AU
```

Implementation Considerations for memberOf

When a new user is provisioned, add the user entry before being assigned to a group.

Referential integrity cannot be guaranteed. If a group is added at the same time as a group is removed, the user can be left in a group that no longer exists. To prevent users being left in a group that does not exist, handle group updates with a single application.

We recommend that you rename groups by performing a delete and an add operation, otherwise, the changed name is not reflected in the memberOf attribute. If this error occurs, a cautionary alarm is issued.

Enable memberOf Attribute with DXmanager

To see which groups an entry belong to, you can enable the memberOf function.

To enable the memberOf attribute with DXmanager

1. In DXmanager, navigate to the Security tab.
2. Add one or more DNs under the memberOf group and user containers. Enter the following settings:

   ```
   set memberof-user-containers = <DN>, ...;
   set memberof-group-containers = <DN>, ...;
   ```

   Important! Both items must be set, otherwise, the DSA produces a critical alarm and shutdown.
3. Save changes and deploy the configuration.

   The memberOf functionality is enabled. To view the configured values, you can use the 'get assoc;' console command.

Note: The memberOf attribute schema is required but you do not need to include it in the entry objectClass. Mark this attribute 'no-user-modification' in the schema as direct updates of memberOf can cause data integrity problems. For more information, see DXHOME/config/schema/sunone.dxc.

Access Controls

The group update is performed with the credentials of the binding user. The DSA triggers the memberOf update and therefore bypasses access controls and also schema checking.

If a separate DSA services the user subtree, the DSA handling the group update requires the 'trust-dsa-triggered-operations' trust flag.

Enable memberOf in an Existing Environment

To enable the memberOf feature in an existing deployment, populate the memberOf data through a dump and reload of all the group entries through the front end. Due to the load this generates, you should perform it during off peak times, or when applications are not accessing Directory.

To enable memberOf in an existing environment

1. Execute the following command to help ensure that entries do not contain memberOf attribute.

   ```
   dxsearch –h{host} –p{port} -b "ou=Users,o=CA,C=AU" "(memberOf=*)" memberOf
   ```

   Entries that contain memberOf attribute are returned. Remove memberOf from any entries that are returned.

2. Retrieve and store groups.

   ```
   dxsearch -h{host} -p{port} -b "ou=Groups,o=CA,C=AU" "(member=*)" member
   objectClass > groups.ldif
   dxsearch -h{host} -p{port} -b "ou=Groups,o=CA,C=AU" "(uniqueMember=*)"
   uniqueMember objectClass >> groups.ldif
   ```

3. Update groups.ldif to remove search summary.

4. Remove groups (Verify that memberOf is not configured).

   ```
   cat groups.ldif | grep "dn: " | awk '{print $2}' | dxdelete -h{host} -p{port}
   ```

5. Enable memberOf functionality and re-init/restart DSAs.

6. Add groups.

   ```
   cat groups.ldif | dxmodify -h{host} -p{port} –a
   ```

Example: Migration

This example migration shows how you can export group and user containers:

```
set memberof-group-containers = <c AU><o CA><ou Groups>;
set memberof-user-containers = <c AU><o CA><ou Users>;
```

How memberOf Updates are Triggered

When a group entry or member is updated, CA Directory can trigger a memberOf Update. In the following description, group entry is a groupOfNames or groupOfUniqueNames, and a member includes a uniqueMember.

The following are the types of updates that can trigger a memberOf update:

- Modify adding one or more DNs to member attribute of group entry
- Modify removing one or more DNs from member attribute of group entry
- Add group entry containing one or more DN member attributes
- Remove group entry containing one or more DN member attributes

When a data DSA recieves a modify request, the following occurs:

1. CA Directory inspects its contents to determine whether all the following conditions are true:
   - The baseObject of the update is subordinate to a DN from the list of configured 'memberof-group-containers'.
   - The baseObject exists for a modify or delete request, and the baseObject does not exist for an add request.
   - The update applies locally.
   - The user performing the update has the appropriate AC to perform the operation.
2. For each member attribute that is subordinate to a DN from the list of configured 'memberof-user-containers':
   a. The request is performed on the user entry, and the group DN is added to or removed from memberOf, and a rollback modify is created.
   b. If the request is successful then a a rollback modify is inserted into rollback list. If an error occurs, a rollback is performed.
3. If memberOf attributes have been updated for all user entries the following occurs:
   a. A group update is performed.
   b. If an error occurs, memberOf updates are rolled back.

Multi-write Replication

Updates of group membership received by multi-write do not trigger memberOf updates. The DSA performing the memberOf population handles replication.

Dynamic Groups and Roles

Dynamic groups and roles work in the same way as static groups and roles (see page 210), except for one major difference: a dynamic group does not store each member DN in its directory entry.

Instead, dynamic groups base the role membership on an LDAP filter.

Each dynamic group entry includes an LDAP search request that is executed when a base-object search is performed on the dynamic group entry. This search finds all the directory entries that satisfy the search filter. These entries are members of the group.

You cannot make the same entry work as both a static and a dynamic role.

Dynamic groups are useful when you know that you often need to change the membership of a group, because there is no overhead involved in maintaining the group data.

If a user is removed from the directory, then that user's entry is not found by the LDAP search when it is next evaluated, so the user is automatically removed from the dynamic group.

Dynamic roles are based on the *dxMemberURL* attribute of the following object classes:

- dxDynamicGroupOfNames
- dxDynamicGroupOfUniqueNames.

You can use dynamic groups to create dynamic roles.

Set Up Groups and Roles

This section describes how to set up static and dynamic groups and roles.

Setting Up Static Groups

To use static groups, you need to do the following:

1. Enable static groups (see page 218).
2. Create a static group entry (see page 218).
3. Add members to the static group entry (see page 219).

   You can then do one or both of the following:

   - Create an application role.
   - Set up static directory roles.

Enable Static Groups

Before you create a static group, you need to prepare the directory.

To enable static groups

1. Stop the DSA.
2. Make the DSA sources the *x500.dxc* schema file supplied with CA Directory. This provides the required object classes.

   To do this, add the following command to the schema file sourced by the DSA:

   ```
   source "x500.dxc";
   ```
3. Start the DSA.
4. Ensure that the DIT contains a subtree in which you can store the group entries.

Create a Static Group

Because a static group is a directory entry, you can create a static group without stopping the DSA.

You should create a special subtree for group entries. This helps you implement roles later.

To create a static group

1. (Optional) Create a new subtree for group entries, if none exists yet.
2. Create an entry in the groups subtree with the object class *groupOfNames* or *groupOfUniqueNames*.

Add a Member to a Static Group

A static group is a directory entry that contains a list of the members of that group.

If the group has the object class *groupOfNames* or *groupOfUniqueNames*, the members are stored in the attribute *member* or *uniqueMember*.

These are multi-valued attributes, and there is no limit to the number of members a static group can have. However, a group with too many entries is very slow to update. We recommend that you use dynamic groups for very large groups.

To add a user to a group, add the user's DN to the *member* or *uniqueMember* attribute.

Example: Add Two Entries to a Static Group

In this example, two users are given the role *AdminFinance*.

```
add-entry-req
    entry = <countryName "AU">
        <organizationName "Democorp">
        <organizationalUnitName "Roles">
        <commonName "AdminFinance">
    contents = {
        (objectClass groupOfNames )
        (member <c au><o Democorp><ou Clerical><cn "Noelene Correa">,<c au><o
    Democorp><ou Finance><ou Budgets><cn "Linda Deacon"> )
    };
```

Setting Up Static Directory Roles

To use static roles, you need to do the following:

1. Set up static groups:
    a. Enable static groups (see page 218).
    b. Create a static group (see page 218).
    c. Add members to the static group entry (see page 219).
2. Enable static roles (see page 220).

    You can then do one or both of the following:

    - Create access control rules for the new role.
    - Set other configuration items for the new role.

Enable Static Roles

Before you can create static roles, you must set up static groups. Ensure that groups are working without error before you set up roles.

A static group becomes a static role when the *set use-roles = true* command is used.

To enable static roles

1. Stop the DSA.
2. Specify the subtree that stores the group entries by including the following commands in the DSA's configuration:

   ```
   set role-subtree = dn;
   set use-roles = true;
   ```

   In this command, *dn* is the distinguished name of the subtree used to store the group entries.
3. Start the DSA.

Disable Static Roles

To disable static roles, enter the following command:

```
set use-roles = false;
```

Setting Up Dynamic Groups

To use dynamic groups, you need to do the following:

1. Enable dynamic groups (see page 221).
2. Create a dynamic group (see page 222).
3. Add members to the dynamic group (see page 222).

   You can then do one or both of the following:

   - Create an application role.
   - Set up dynamic directory roles.

Enable Dynamic Groups

Dynamic roles are based on the *dxMemberURL* attribute of the following object classes:

- dxDynamicGroupOfNames
- dxDynamicGroupOfUniqueNames

You can add these attributes to a *groupOfNames* or *groupOfUniqueNames* object class respectively so that *dxMemberURL* can be included.

To enable dynamic groups

1. Stop the DSA.
2. Add the following commands to the DSA's settings:

   ```
   clear dynamic-group;
   ```

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

   For example:

   ```
   set dynamic-group GROUP = {
   objectclass = dxDynamicGroupOfNames
   url-attr = dxMemberURL
   member-attr = member
   };
   ```

3. Start the DSA.
4. Ensure that the DIT contains a subtree in which you can store the roles entries.

   For instance, the examples use the following subtree (see page 226):

   ```
   c=AU,o=Democorp,ou=Groups
   ```

Create a Dynamic Group

Because a dynamic group is a directory entry, you can create a dynamic group without stopping the DSA.

You should create a special subtree for group entries. This helps you implement roles later.

To create a dynamic group entry

1. (Optional) Create a new subtree for group entries, if none exists yet.
2. Create an entry in the groups subtree with the following information:
   - Object class: *groupOfNames*
   - LDAP search filter: Include this in the attribute *dxMemberURL*.

Add a Member to a Dynamic Group

To add a member to a dynamic group, include information in the user's entry so that the filter for that role is satisfied.

Example: Create a *manager* Group

You have created a dynamic group that uses the following filter:

```
ldap://o=user,c=AU??sub?(position=manager)
```

To assign the manager group to a user, do the following:

1. Add the *position* attribute to the user's entry.
2. Add the *manager* value to the position attribute.

The next time that the *manager* group is evaluated, this user is a member of the group.

View a Dynamic Group's Configuration

To view the dynamic group configuration in use in a DSA, use the following command from the DSA console:

```
get dynamic-groups;
```

This produces a list of dynamic groups in use.

Example: Output from the get dynamic-groups Command

```
************** GROUP **************
Group object class : dxDynamicGroupsOfNames
Group Search URL : dxMemberURL
Member to Append : member
```

Setting Up Dynamic Directory Roles

To use dynamic roles, you need to do the following:

1. Set up dynamic groups:

   a. Enable dynamic groups (see page 221).

   b. Create a dynamic group (see page 222).

   c. Add members to the dynamic group (see page 222).

2. Enable dynamic roles (see page 223).

3. Create a dynamic role (see page 224).

   You can then do one or both of the following:

   - Create administrative limits for the new role.
   - Create access control rules for the new role.

Enable Dynamic Roles

Before you can create dynamic roles, you must set up dynamic groups. Ensure that groups are working without error before you set up roles.

To enable dynamic roles

1. Stop the DSA.

2. Add the following commands to the DSA's settings configuration file, in DXHOME/config/settings:

   ```
   set role-subtree = DN;
   set use-dynamic-roles = true;
   ```

   In this command, *DN* is the distinguished name of the group's subtree.

3. Start the DSA.

Create a Dynamic Role

For each dynamic role, you need to create an entry in the roles subtree using one of these auxiliary object classes:

dxDynamicGroupOfNames

This object class contains an attribute *groupOfNames*, which you can use to store the DN.

dxDynamicGroupOfUniqueNames

This object class contains an attribute *groupOfUniqueNames.*

All role entries must be stored in the same subtree.

To create a dynamic role

1. Add a value to the *dxMemberURL* attribute of a dynamic group containing a search filter in LDAP URL form:

   `ldap:///base-dn??scope?filter`

   base-dn

   Specifies the base object for the filter search.

   scope

   (Optional) One of the following:

   sub

   Specifies that the filter searches the entire subtree below the base DN.

   base

   (Default) Specifies that the filter returns just the DN.

   one

   Specifies that the filter searches one level below the base DN.

   filter

   (Optional) Defines the LDAP search filter, for example:

   ```
   (|(group=teachers)(group=students))
   ```

2. Save the change to the entry.

   The role is applied to members when they next log in to the directory.

Example: A Dynamic Role Entry

This example shows a dynamic group entry that is used as a role.

The entry is shown in LDIF format:

```
dn: cn=Manager,ou=Groups,o=Democorp,c=AU
objectClass: groupOfNames
objectClass: dxDynamicGroupOfNames
objectClass: top
cn: Manager
dxMemberURL:: bGRhcDovLy9jPVFVSz9zdWI/AHNuPOR1bWVsZWXvssUpIA=0
```

The DN in the *dxMemberURL* attribute is encoded, because of the attribute's syntax. The unencoded value is as follows:

```
ldap:///o=Democorp,c=AU??sub?(position=manager)
```

In this URL, the search's base object and scope are ignored.

Disable Dynamic Roles

To disable dynamic roles, enter the following command:

```
set use-dynamic-roles = false;
```

Role-Based Configuration

After you have set up a group, you can add some specified attributes that restrict the searches for group members. A group used to provide restrictions in this way is called a role.

This works for both static and dynamic directory groups. You can use role-based configuration in both local and distributed directories.

For a group to hold these attributes, it needs to have the *dxRoleBasedConfig* auxiliary object class. *dxRoleBasedConfig* lets the group entry contain attributes that define the following types of restrictions:

Operational limits

You can add the following attributes to a role:

- *dxTimeLimit*—This attribute specifies the maximum duration for a search.
- *dxSizeLimit*—This attribute specifies the maximum number of entries that a search can return.

For more information about operation limits, see Limit Operations (see page 302).

Search profiles

You can add the attribute *dxAllowSearch* to a role. This attribute specifies the name of a search profile. Search Profiles (see page 311) are defined by the *set allow-search* command.

You can also use a group to provide role-based access controls. For more information, see Access Controls (see page 153).

More information:

Apply Operational Limits to a Role (see page 307)
Assign a Role to an Access Control Rule (see page 164)

Examples: Static Groups and Roles in Democorp

The following examples show how to set up static groups and roles in Democorp, one of the sample directories provided with CA Directory. You can use these examples as training exercises.

Example: Enable Static Groups in Democorp

Before you create a static group, you need to prepare the directory.

The Democorp directory does not contain any groups. Use these instructions to prepare the directory for static groups.

To enable static groups

1. Stop the Democorp DSA by entering the following command at a command prompt:

   ```
   dxserver stop democorp
   ```

2. Source the x500.dxc schema file in the Democorp configuration, as follows:

   a. Look in the DXHOME/config/servers directory for the *democorp.dxi* file, and open it in a text editor.

      This is the Democorp DSA's initialization file.

   b. Find the following schema file statement:

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

      This means that when the Democorp DSA starts, it sources the schema files listed in *samples.dxg*.

   c. Make the *samples.dxg* file writable.

   d. Open *samples.dxg* in a text editor, and add this line to the file:

      ```
      source "x500.dxc";
      ```

   e. Close and save *samples.dxg*.

3. Make the DSA sources the *x500.dxc* schema file supplied with CA Directory. This provides the required object classes.

   To do this, add the following command to the schema file sourced by the DSA:

   ```
   source "x500.dxc";
   ```

4. Start the Democorp DSA by entering the following command at a command prompt:

   ```
   dxserver start democorp
   ```

5. Create a new subtree to store the group entries, as follows:

   a. Open JXweb and connect to the Democorp DSA.

   b. Select the DEMOCORP entry in the tree on the left, and then click New.

   c. Enter *ou=Groups* in the RDN for New Entry field, click Submit twice, and then click OK.

   Any group entries should be created under this entry.

Example: Create a Static Group in Democorp

This example shows how to create a new static group entry in the Democorp directory by loading it from an LDIF file.

This group lists the first aid officers in Democorp. It is already populated with two group members.

To add a static group to the Democorp directory

1. Save the following in a text file named *firstaid.ldif*:

   ```
   version: 1

   dn: cn=First Aid Officers,ou=Groups,o=DEMOCORP,c=AU
   objectClass: groupOfUniqueNames
   objectClass: top
   cn: First Aid Officers
   uniqueMember: cn=Craig LINK,ou=Administration,ou=Corporate,o=DEMOCORP,c=AU
   uniqueMember: cn=Vivienne LEVER,ou=Administration,ou=Corporate,o=DEMOCORP,c=AU
   ```

2. Open a command prompt and change to the directory in which you saved the LDIF file.

3. Load the entry into the Democorp directory, using the following command in the command prompt:

   ```
   dxmodify -a -h hostname -p portnumber -f firstaid.ldif
   ```

   -h *hostname*

   Specifies the name of the computer on which the DSA is running.

   -p *portnumber*

   Specifies the port number of the DSA. By default, the Democorp sample DSA uses port 19389.

   The new entry First Aid Officers can now be used as a static group. You can now add or delete members from this group.

Example: Enable Static Roles in Democorp

This example shows how to enable static roles in Democorp. To enable roles, you need to change the DSA's settings, which means that you have to stop and start the DSA. These instructions also describe how to identify which settings file is used by the Democorp DSA.

To enable static roles in Democorp

1. Identify which settings file is used by Democorp, as follows:
   a. Open Democorp's initialization file in a text editor.
   b. Find the following schema file statement:

      ```
      source "../settings/dxmanager.dxc";
      ```

      This means that when the Democorp DSA starts, it uses the settings in *settings/dxmanager.dxc*. You need to add the role commands to this file.
2. Stop the Democorp DSA.
3. Add the commands to enable static roles to the Democorp settings file. To do this:
   a. Ensure that the settings file is writable, and then open it in a text editor.
   b. Add the following lines to the settings file:

      ```
      #roles
      set role-subtree = <c AU><o Democorp><ou Groups>;
      set use-roles = true;
      ```

   c. Close and save the settings file.
4. Start the DSA.

   The Democorp DSA can now use static groups as roles. You can now set limits or access controls for any such groups you have created.

Examples: Dynamic Groups and Roles in Democorp

The following examples show how to set up dynamic groups and roles in Democorp, one of the sample directories that comes with CA Directory. You can use these examples as training exercises.

Example: Create a Dynamic Group in Democorp

This example shows how to create a new dynamic group entry in the Democorp directory using JXweb.

This dynamic group includes entries in Democorp that have the word *engineer* in the *description* attribute.

To create a dynamic group in Democorp

1. Open JXweb and connect to the Democorp DSA.
2. Navigate to the Groups subtree and click New.
3. Enter and select the following information:
   - **Parent DN:** ou=Groups,o=DEMOCORP,c=AU
   - **RDN for New Entry:** cn=Engineers
   - **Selected:** *groupOfNames* and *dxDynamicGroupOfNames*
4. Click Submit.

   The new entry is created. You now need to add the LDAP filter to the *dxMemberURL* attribute.
5. Display the attributes without values by clicking the arrow beside *List of attributes without values*.
6. Click the Edit icon next to the *description* attribute, enter the following filter, and then click Modify:

   ```
   ldap:///o=Democorp,c=AU??sub?(description=*engineer*)
   ```

   The new entry Engineers can now be used as a dynamic group.

Example: Enable Dynamic Roles in Democorp

This example shows how to enable dynamic roles in Democorp. To enable roles, you need to change the DSA's settings, which means that you have to stop and start the DSA. These instructions also describe how to identify which settings file is used by the Democorp DSA.

To enable dynamic roles in the Democorp DSA

1. Identify which settings file is used by Democorp. To do this:
    a. Open Democorp's initialization file in a text editor.
    b. Find the schema file statement. By default, this statement is :

    ```
    source "../settings/dxmanager.dxc";
    ```

    This means that when the Democorp DSA starts, it uses the settings in *settings/dxmanager.dxc*. You need to add the role commands to this file.
2. Stop the Democorp DSA.
3. Add the commands to enable dynamic roles to the Democorp settings file. To do this:
    a. Ensure that the settings file is writable, and then open it in a text editor.
    b. Add the following lines to the file:

    ```
    #roles
    set role-subtree = <c AU><o Democorp><ou Groups>;
    set use-dynamic-roles = true;
    ```

    If this file already includes the *set use-dynamic-roles = true* command, you can leave it there. A DSA can contain both static and dynamic roles.
    c. Close and save the settings file.
4. Start the DSA.

    The Democorp DSA can now use dynamic groups as roles. You can now set limits or access controls for any such groups you have created.

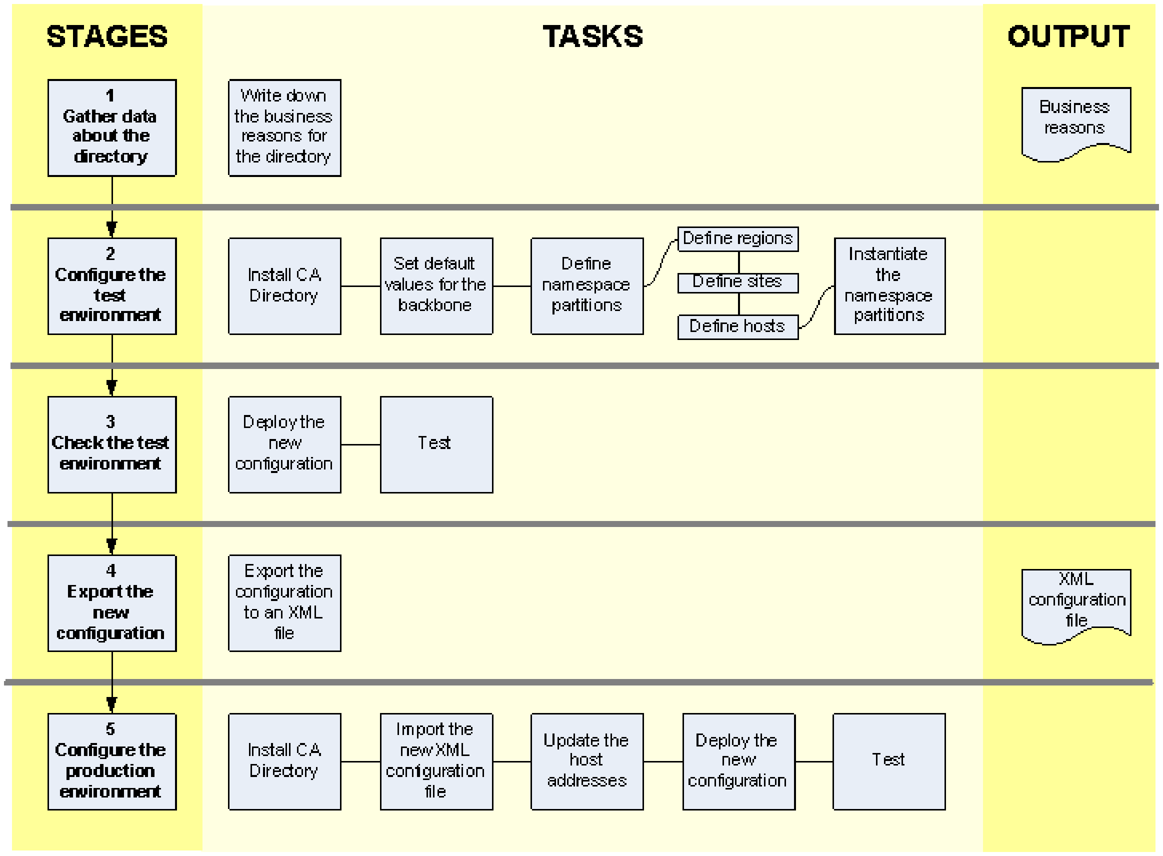
Chapter 12: Use DXmanager to Create and Modify the Backbone

This section contains the following topics:

Overview of the Process to Create a New Directory Backbone

To create a new directory backbone with CA Directory, use DXmanager.

The following flowchart shows the steps to create a new directory backbone. Each step is described fully in the following sections.



STAGES
TASKS
OUTPUT
1 Gather data about the directory
Write down the business reasons for the directory
Business reasons
2 Configure the test environment
Install CA Directory
Set default values for the backbone
Define namespace partitions
Define regions
Define sites
Define hosts
Instantiate the namespace partitions
3 Check the test environment
Deploy the new configuration
Test
4 Export the new configuration
Export the configuration to an XML file
XML configuration file
5 Configure the production environment
Install CA Directory
Import the new XML configuration file
Update the host addresses
Deploy the new configuration
Test

Gather Data about the Directory

Before you start creating the directory configuration in the test environment, you should gather information about the requirements for the directory.

Record the Business Reasons for the Directory

Before you design the new configuration, ensure that you are familiar with the directory's business reasons, and all challenges that the previous configuration had to overcome.

Review any design notes or documentation on the current configuration so that you are aware of any special requirements implemented by the design.

Here are some possibilities:

- Was query streaming used? This is no longer necessary.
- Were there specific security requirements?
- Was a specific schema required for any applications (for example, CA SiteMinder)?
- Was a password policy defined?
- Were DSAs in the same or different regions?
- Were load-share groups or other methods of restricting load-sharing used?

Define the Performance Requirements

Before you design a directory, you should work out the performance and scalability requirements.

The level of performance you require is defined by the applications that use the directory. If the applications have a service level agreement (SLA), you can use that as a start.

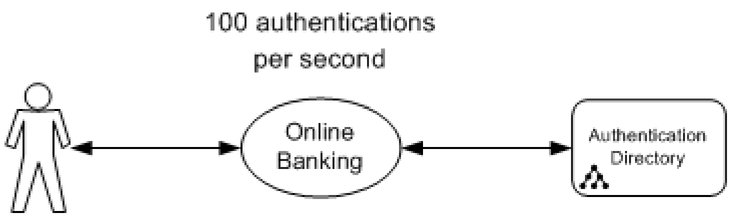
The performance of a directory is measured in the number of searches and updates performed per second.

Translate the Application's Requirements

You must translate the application's requirements into directory terms. Directory performance is measured in operations per second.

Example: Translate the Application Requirements into Directory Terms

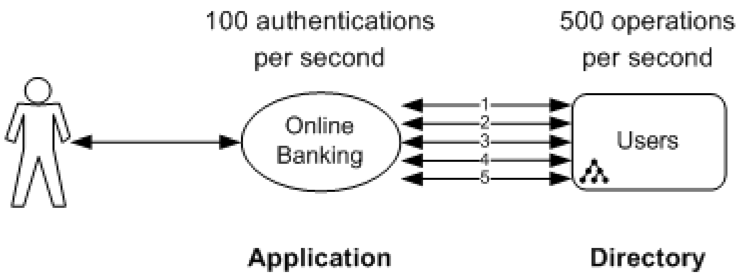
An online banking application uses a directory as a repository for user authentication data. The application has an SLA (service level agreement) of 100 authentications (user logins) per second, as shown in the following diagram:



This SLA is helpful for working out the performance requirement for the directory, but it is not enough to base a directory design on.

In this example, for each authentication, this application actually performs five operations on the directory, including a search to see if the user entry is present and an update to the “last login time” value.

This means that the performance requirement for the directory is actually 500 operations per second, as shown in the following diagram:



Analyze All Applications

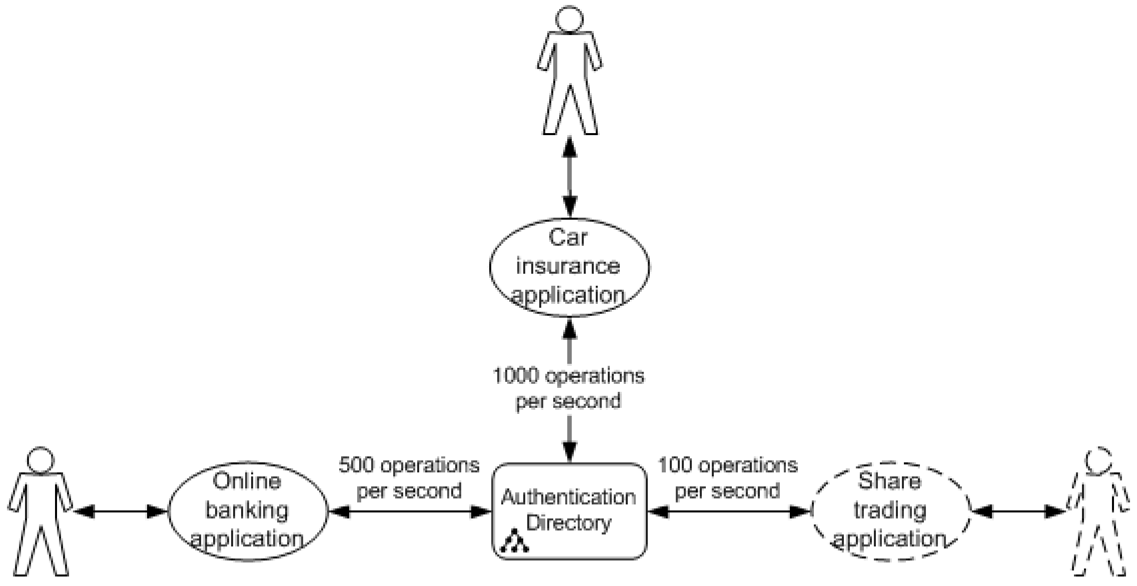
If the directory is used by many applications, you must use the *sum* of the applications' requirements to work out the actual requirement for the directory.

If you work out the requirement for an individual application only, the directory may not perform as required.

Example: Performance with Many Applications

The following diagram shows a single authentication directory that is currently used by two applications (online banking and car insurance).

In this example, the third application (share trading) is new, and the directory designer is currently checking the performance of the directory with this new application.


Car insurance application
1000 operations per second
Online banking application
500 operations per second
Authentication Directory
100 operations per second
Share trading application

Even though the new application only requires the directory to respond to 100 operations per second, the directory is also servicing requests from the other applications.

This directory must be stress-tested under the full load of 1600 operations per second.

Configure the Test Environment

You now need to use DXmanager to set up the configuration for the new directory backbone. Do this preparation in the test environment, and later move the new configuration file to the production environment.

The test environment should mimic the production environment as closely as possible. This includes hardware, software, and data.

We recommend that you use short descriptive names for namespace partitions, regions, sites, and hosts. DXmanager uses these names to label each part of your directory backbone. The DXmanager maps have limited space, so we recommend that you use short names to make the maps as clear as possible.

To configure the test environment

1. Install CA Directory on the test environment.
2. Start DXmanager (see page 242).
3. Configure a new directory backbone (see page 243).
4. Add any more regions, sites, or hosts (see page 246).
5. Add any more namespace partitions (see page 247).
6. Add more DSAs (see page 249).
7. Deploy the configuration (see page 244).

Modifying CA Directory Configuration to Use DXmanager

CA Directory lets you use DXmanager to manage the configuration of the entire backbone.

Before You Begin

We recommend that you read How CA Directory Works (see page 25) before you upgrade to DXmanager, to find out what has changed.

Use a Test Environment

We strongly recommend that you use a test environment to create the new configuration that uses DXmanager.

You can then copy the new configuration file to the production environment.

The benefits of using this method are as follows:

- You can test the upgrade procedure without affecting your production environment.
- You can learn about the new configuration system without affecting the production environment.
- You can take the time to design a simpler configuration than you have today.

This should result in minimal disruption and greater confidence when you upgrade your production environment.

Complex Configurations

Your directory may use a complex configuration. Complex configurations rely on obsolete features of the old-style knowledge files.

If you are using a complex configuration, you must resolve it before you can upgrade CA Directory. You may need help in translating your configuration.

Knowledge Hiding

Knowledge hiding is sometimes used to deliberately give some DSAs no knowledge of other DSAs. A DSA cannot route a query to a DSA that it has no knowledge of.

The disadvantage of knowledge hiding is seen during failover: router DSAs need to know all DSAs that serve each namespace partition.

Knowledge hiding is not required if you use DXmanager, as is explained in the following table:

Reason to Use Knowledge Hiding	**Description**	**New Method using DXmanager**
Reduce the number of edits to configuration files	Knowledge hiding reduced the amount of knowledge information stored in configuration files, which meant that after a configuration change, fewer files had to be edited.	Knowledge files are not used—You now use DXmanager to manage the whole directory backbone.

Reason to Use Knowledge Hiding	Description	New Method using DXmanager
Prevent long searches from delaying other queries	Knowledge hiding permitted the use of reporting applications that send long search queries.	Use role-based limits.

How to Modify Your Directory Configuration to Use DXmanager

These instructions assume that you do not need to redesign the directory backbone, and that you need only to convert your existing design into the new format.

To modify your CA Directory installation to use DXmanager, follow these steps:

1. Gather data about the directory:
   a. Record the business reasons for the directory (see page 234).
   b. Analyze all applications (see page 236).
   c. Copy the configuration files from the production environment (see page 240).
2. Install CA Directory in the test environment

   See the Installation Guide for what to install on each computer.
3. Configure the test environment:
   a. Start DXmanager (see page 242).
   b. Configure a new directory backbone (see page 243):
   c. Add any more regions, sites, or hosts (see page 246).
   d. Add any more namespace partitions (see page 247).
   e. Add more DSAs (see page 249).
   f. Deploy the configuration (see page 244).
4. Check the test environment. (see page 240)
5. Export the new configuration to an XML file (see page 240).
6. Upgrade the production environment:
   a. Back up the directory's configuration and data.
   b. Install CA Directory in the production environment
   c. Start DXmanager (see page 242).
   d. Import the new XML configuration (see page 244).
   e. Update the host addresses (see page 244).
   f. Deploy the configuration (see page 244).

7. Test the production environment to determine that CA Directory is functioning as you expect and that the upgrade is complete.

   If there is a problem, you can roll back the production environment to the previous configuration and resolve the problem using the test environment.

Copy the Configuration Files from the Production Environment

At this stage, DXmanager supports old server and knowledge settings but not access control and schema settings. Therefore, you need to copy custom access control and schema settings into the appropriate config directories once you set up the new CA Directory environment (see page 237). You can use all other files as a reference when you use DXmanager to do the initial setup.

To copy the configuration files from the production environment

1. Create a folder on the directory management server to store the old configuration files.
2. Create a subfolder for each computer in the production environment.
3. Copy the entire DXHOME/config directory from each computer in the production environment into the corresponding folder.

   You should now have a single folder that contains one subfolder for each host, each of which contains the old configuration files from that host.

Check the Test Environment

You should test the newly configured directory backbone to ensure that it performs as you expect.

We recommend the following tests:

- Load some test data into the directory, and then set up the test applications.
- Perform some application tasks to verify that they run correctly with the new directory backbone you have set up.

Export the New Configuration to an XML File

To export the configuration to an XML file, use DXmanager's export function.

This creates an XML file that contains the new configuration. You can later import this file using DXmanager in the production environment.

Configure the Production Environment

After you have created the new directory configuration in the test environment, you can move that configuration to your production environment.

To configure the production environment

1. Install CA Directory on the production environment.
2. Create new DXmanager user accounts (see page 53).
3. Start DXmanager (see page 242).
4. Import the new XML configuration (see page 244).
5. Update the host addresses (see page 244).
6. Deploy the configuration (see page 244).
7. Test the production environment to determine that CA Directory is functioning as you expect and that the upgrade is complete.

   If there is a problem, roll back the production environment to the previous configuration and executables. You should then resolve the problem using the test environment.

How to Use DXmanager to Manage the Configuration

Start DXmanager

DXmanager is a web application that lets you create, configure, monitor, and control your directory backbone. You can connect to DXmanager from any computer that has a web browser and a connection to the computer on which DXmanager is installed.

To use DXmanager, you need to log in. A superuser account is created during installation. You can use this account to log in and create more user accounts.

To start DXmanager

1. Start DXmanager as follows:
   - Windows: Click Start, All Programs, CA, Directory, DXmanager.
   - All platforms: Open a web browser and go to the following URL:

     https://*hostname*:*portnumber*/dxmanager

     hostname

     Specifies the name of the directory management server.

     portnumber

     Specifies the port number of CA Directory Web Server on which DXmanager runs.

     Default: 8443

   The Login page opens in the browser.
2. Enter the following information to log in as the superuser:
   - **User Name:** *DXmanager*
   - **Password:** The password that was created during the installation
3. Click Log In.

   DXmanager opens in your browser.

Configure a New Directory Backbone

When you open DXmanager for the first time, you can run a wizard to help you create a simple backbone.

The DXmanager wizards include some default values. We recommend that you use these default values unless you are sure that your backbone requires a different value.

To configure a new directory backbone

1. Log in to DXmanager as a user with the Change or Superuser role. These roles give you permission to change the directory backbone's configuration.

   DXmanager opens, showing a page that links to the following items:

   Single DSA Backbone

   Opens a wizard that helps you enter the information required to create a simple backbone with one host that contains a single DSA.

   Advanced

   Opens DXmanager in Edit mode, so you can create the configuration without the wizard.

2. Click the Single DSA Backbone link.
3. In the first page, define default values for the whole backbone (see page 245).
4. In the second page, define namespace partitions (see page 247).
5. Define the network topology in the following order:
   a. Define the regions.

      Most directory backbones require only a single region. You should define more than one region only if areas of your network have high latency, slow links, or low bandwidth.

   b. Define the sites.

      A site is a collection of hosts, over which router DSAs can share the load of requests. Each site defined in DXmanager is likely to correspond to an actual data center.

   c. Define the hosts.
6. Click Configuration, Deploy to update the hosts with the new configuration.

More information:

The Backbone Concept in DXmanager (see page 40)
How to Set Up the Sample Directories (see page 368)

Import the XML Configuration

You can import the entire XML configuration into DXmanager.

This is useful if you have set up a test environment to develop the configuration. You can later import the configuration into the production environment.

Update the Host Addresses

When you created the configuration in the test environment, the network addresses of the hosts were included in the backbone configuration. You need to change these network addresses to match your production environment.

To update the host addresses

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Change the network address for each host from the address of the test environment to the network address for the production environment.

   Ensure that you change every host.

Deploy the Configuration

Deploying is the process of transferring the XML configuration file from DXmanager to the DSA's host, and then reinitializing the DSAs.

To deploy the configuration

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click Configuration, Deploy to update the hosts with the new configuration.

   The updated configuration file is sent to all hosts. Any new DSAs are created automatically. Any existing DSAs are reinitialized.

How to Modify the Backbone

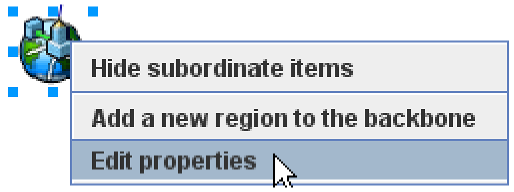
You can modify any aspect of a directory backbone. When you use DXmanager to change the configuration, the XML file on the DXmanager server is updated. You can then deploy the new configuration to the hosts in the backbone.

Define Default Values for the Whole Backbone

The DXmanager wizards include some default values. We recommend that you use these default values unless you are sure that your backbone requires a different value.

To define default values for the whole backbone

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Topology map.
4. Right-click on the Backbone icon, and then select Edit Properties, as shown in the following example:



5. In the Edit Backbone Default dialog, specify the configuration values for the entire backbone as defaults.
6. Click OK when you are finished.
7. Click Configuration, Deploy to update the hosts with the new configuration.

Note: If this is the first time you have run DXmanager and you are using the Create a Backbone wizard, enter these default values in the first page of the wizard, and then click OK and deploy the configuration.

Create a New Region, Site, or Host

The *network topology* describes the hosts on which the directory backbone runs, and the quality of network connections between the hosts. The quality of connections is represented by *sites* and *regions*.

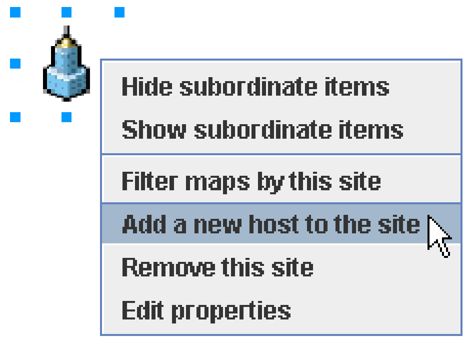
When you want to add another computer to the directory backbone, you should add a new host in the directory configuration. Similarly, you can add new sites or regions if your company has acquired data centers.

We recommend that you give each host, region, and site a short descriptive name.

To create a new region, site, or host

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Topology map.
4. Right-click the level above the item you want to create, and then select the option to add a new item.

   The following example shows the option for creating a new host by right-clicking on a site:

5. Enter the information about the new region, site, or host in the dialog, and click OK when you are done.

   The Topology map now displays the new region, site, or host.

6. Click Configuration, Deploy to update the hosts with the new configuration.

More information:

Network Topology in DXmanager Terms (see page 41)

Create a Namespace Partition

A *namespace partition* is a sub-section of the directory information tree.

You need to create a namespace partition before you can create a DSA that serves the partition.

Note: Decide if this namespace partition is to be served by a router DSA or a data DSA. One namespace partition cannot be served by both types of DSA.

To create a namespace partition

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Namespace map.
4. Right-click on any node and select *Add a new partition*.
5. Enter the following details:

   Partition name

   Specifies the name of the namespace partition that you are creating. This name is used in DXmanager displays.

   Prefix

   If this namespace partition is to be served by a data DSA, then this specifies the context prefix of the DIT that the DSA publishes that it serves. If *Native prefix* is also set, then this becomes the logical prefix rather than the physical prefix and prefix mapping is done in the server.

   If you plan to use a router DSA to serve this namespace partition, then leave this blank.

   SNMP port

   Specifies the SNMP port for this partition. This is the port the DSA will use to send SNMP traps. This is needed by the DXmanager to be able to monitor DSAs.

   Port

   Specifies the port associated with this namespace partition. The DSAs created when you deploy this namespace partition to a host uses this port as their listen address.

   Console port

   Specifies the port on which the DSA accepts connections from the local computer only. If this is not specified, there is no console for the DSA.

   This is automatically populated when you enter the Port value.

Remote console port

Specifies the port on which the DSA accepts connections from remote computers.

This is automatically populated when you enter the Port value.

Datastore size

Specifies the datastore size in MB.

6. When you are satisfied with the settings, click OK.

   The Maps tab appears, showing the Namespace map.

   You can now assign a DSA to serve this namespace partition.

7. (Optional) Save this version of the configuration, as follows:

   a. Click Configuration, Save.

   b. Enter a comment that describes the current state of the configuration, and then click OK.

Create a DSA

To create a new DSA, you tell DXmanager the host, the namespace that the DSA will serve, and the type of DSA you want DXmanager to create.

To create a DSA

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Topology map.
4. Right-click the host that will hold the new DSA, select *Instantiate a new partition to this host*, and then select one of the DSA types:
   - As a Router DSA
   - As a Data DSA
   - As a third-party X.500 DSA
   - As a third-party LDAP DSA

   The Instantiate dialog opens.
5. Do the following in the dialog:
   a. Select the namespace partition that the new DSA will serve, and then click Next.

      Note: A namespace cannot be served by both a data DSA and a router DSA, so if it is already served by one of these sorts of DSA, the other option is unavailable.

   b. Choose the network address for this DSA. If the host has a single address, you can leave this unchanged.

   c. Click OK when you are done.
6. Click Configuration, Deploy to update the hosts with the new configuration.
7. Do the following on the host that contains the new DSA:
   a. Edit the new initialization file (DXHOME/config/servers/*dsaname*.dxi) to source any customized configuration files. These files can configure logging, schemas, settings, limits, and access controls.

   b. Create the datastores required for the new DSAs.

   c. Load or synchronize each DSA with other DSAs that serve the same namespace partition.

Note: You can also create a new DSA by right-clicking on a partition in the Namespace map.

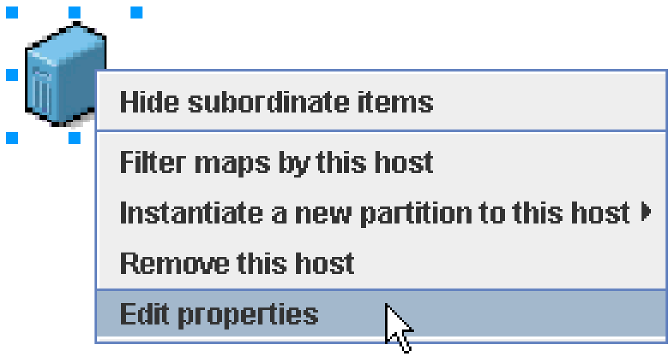
Edit All DSAs on a Host

If you want to change a configuration item for all DSAs on a host, change it at the host level. This change is inherited by all DSAs that do not have a different value set.

To edit all DSAs on a host

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Topology map.

   Right-click the host you want to modify, and then select the *Edit properties* option, as follows:



4. Enter the new configuration information, and click OK when you are done.
5. Click Configuration, Deploy to update the hosts with the new configuration.

View the Properties of Part of the Backbone

DXmanager lets you view the properties of any part of the backbone.

To view the properties of part of the backbone

1. Log in to DXmanager.
2. Click the Maps tab, and then display either the Namespace or Topology map.
3. Right-click any item in the map, and then select View Properties

   The dialog that appears lists the properties of the item you selected.
4. Click Close when you are finished viewing the properties.

Chapter 13: Monitor the Backbone

This section contains the following topics:

Dashboard Tab

The Dashboard tab in DXmanager shows a high-level summary of the health and load of the directory backbone.

The top of the Dashboard area shows the following messages areas:

Alerts

Notifies you of any new alerts. Look at the Alerts tabs to read the messages.

Configuration

Shows whether another user is editing the configuration. Only one user can edit the configuration at a time.

Below the messages areas, the Dashboard area includes the following graphs:

Status graph

Shows the overall status of each region. The possible statuses are running, stopped, partially running, and unknown.

Fullest Multiwrite Queue graph

Shows the size of the largest multiwrite queue in each region.

Total Load graph

Shows the number of operations per second for the entire directory, during the last minute.

Total Searches graph

Shows the number of searches per second for the entire directory, during the last hour.

Maps Tab

DXmanager helps you monitor the health of your directory. You can easily see the current health of every part of the entire backbone, and you can also look at historical data to identify events and patterns.

DXmanager helps you locate the source of any health problems, and work out their severity. You can then quickly take action to solve the problems.

DXmanager can tell you these things:

- Which DSAs and hosts are under the most load
- The nature of that load: searches, updates, compares, and so on.
- The load profile from each application
- When the peak load occurs, and its size
- When the lowest load occurs, such that maintenance outages can be scheduled

How the Maps Show Health

DXmanager lets you monitor the health of your directory in the Namespace and Topology maps.

In each of these maps, each part of the directory is shown as an icon. For example, a site is shown with the following icon:

Every icon also has a small status indicator, as shown in the following table, in order of increasing severity:

Status Indicator	Description
🟩	Running
⊘	Stopped
◐	Some running, and some stopped
?	Unknown

For DSAs, the status indicator shows whether the DSA is running, stopped, or cannot be contacted. For example, a data DSA that is running is shown with the following icon:

For all other parts of the backbone, the status indicator shows a roll-up of the statuses of the DSAs under it. For example, if a site includes some DSAs that are running and some that are stopped, the site is shown with the following icon:

If any DSA has the status *unknown*, all higher sections also show the status *unknown*.

Links between icons can be shown as dashed or solid lines. Dashed lines mean there is no current connection between the icon and its higher-level host, site, region or backbone. A solid line means at least one DSA is running.

How the Topology Map Shows Load

The Topology map shows each regions, site, host, and DSA as a separate node.

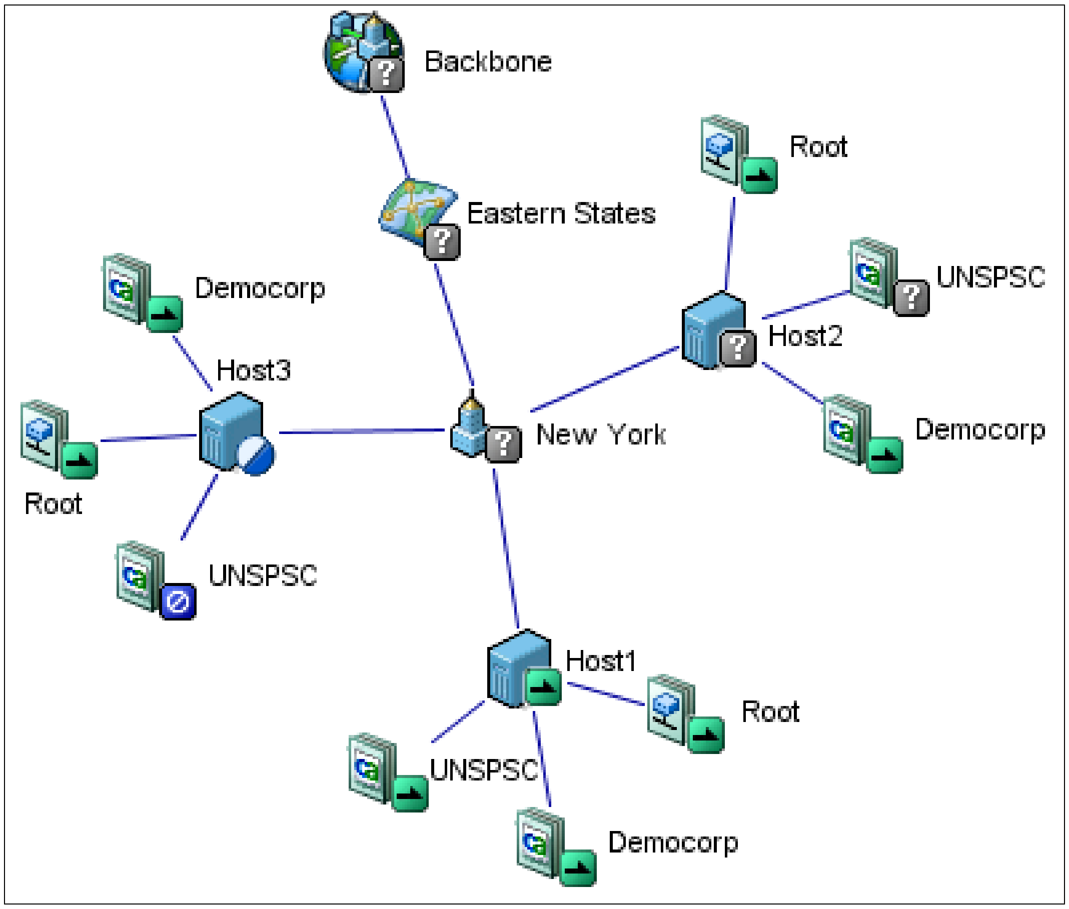
The connections between these nodes are shown as thin lines when there is no load of operations flowing between them.

When a DSA is under load, the connections between topology units swell to show the load, as shown in the following screenshot:

You can hover the cursor over a link to show the percentage of the total load on that link.

Example: Topology Map Showing Health and Load

The following topology map shows the sample directory described in How to Set Up the Sample Directories (see page 368):



The icons show the status of each DSA, and the blue lines show the load on each DSA. Thicker lines show greater load.

The status of the entire backbone is derived using the following steps:

- On Host1, all DSAs are running. The host icon also shows the green icon, because all of its DSAs are running.
- On Host2, two DSAs are running, but the status of the third is unknown. The host shows the worst status, which is *unknown*.
- On Host3, two DSAs are running and one is stopped. The host icon shows the blue-and-white status icon, to show that some of its DSAs are running and some are stopped.
- The site icon shows the worst status of the hosts in that site, which is *unknown*.
- The region has only one site, and the backbone has only one region, so those icons also show the gray icon.

In this way, the Topology map reveals that this directory backbone is in poor health. The DSA with the status of *unknown* must be fixed.

Charts Tab

The Charts tab lets you look at historical data to identify events and patterns.

Note: There is a two to six minute delay for updated data to appear in the charts.

The default chart shows line graphs of operations over time over the last hour.

You can export the data into a CSV file for later processing.

Alerts Tab

Alerts are any events that the user should be made aware of. In this version of DXmanager, the alerts are harvested from the alarm logs.

There are three sources of alerts, as follows:

- DXmanager—Information about DXmanager's communication with DXadmind
- DXadmind—Information about how DXadmind processes DXmanager's requests
- DXserver—Alarms from DSAs.

Alerts can be displayed in these ways:

Alerts tab

All alerts are displayed on the Alerts tab in DXmanager.

Email messages

You can also configure DXmanager to send email notifications containing alerts as they are detected. You can filter which types of alerts are sent to each address.

Other Monitoring Methods

DXmanager is the best method for routine monitoring. However, the following other monitoring tools are each useful in different situations:

- DSA console (see page 257)
- Log files (see page 256)
- Traces (see page 261)

Monitoring with Log Files

The *logs* contain output from a DSA, including its trace, diagnostics, warnings, and alarms.

Note: We recommend that you use only the logs that are actually required. Because a DSA has to write to the log files, keeping too many log files open can reduce performance.

Monitoring with the DSA Console

You can use the DSA console to check the operational parameters and DSA usage with management commands.

To monitor with the DSA console

1. Use Telnet to connect to the DSA that you want to monitor.
2. Use the following commands to do these tasks:
   - View the number and type of operations performed by the DSA:

     ```
     get stats;
     ```

   - View current communication settings on a DSA:

     ```
     get stack;
     ```

   - View current DSP connections to other DSAs:

     ```
     get online-dsas;
     ```

   - View current user connections to the DSA:

     ```
     get users;
     ```

   - View the names of the currently enabled log files:

     ```
     get log;
     ```

Types of Logs

CA Directory lets you set up the following types of log for each DSA:

Alarm log

The *alarm log* contains all alarms. *Alarms* are reports of critical events that should be monitored. This is not dependent on the tracing level or whether the DSA console is open.

This is the only log that cannot be closed. It is always open when a DSA is running.

This log has a default name of *dsa-name*_alarm.log.

Alert log

An *alert log* contains all authentication errors and account suspensions. It can be used to show attempts at unauthorized access to the DSA. This is not dependent on the tracing level.

Certificate log

A *certificate log* contains a summary of operations that involve certificates or CRLs. This includes all add and modify operations that include a *userCertificate*, *caCertificate*, or *certificateRevocationList* attribute. In addition, any read request, search request, or search filter that returns one of these attributes is recorded.

This is not dependent on the tracing level.

Connection log

A *connection log* contains a line for each successful connection made, and each released connection. This is not dependent on the tracing level.

The connection log is time-stamped and date-stamped, and a new one is written daily.

Diagnostic log

A *diagnostic log* contains a list of operations that the DSA has rejected, for whatever reason. This includes the operation, the DN of the affected entry, and a diagnostic message. Use this file to debug applications.

If diagnostic tracing has been enabled, this output is also sent to the trace log.

Query log

A *query log* contains detailed information about every operation, including a time and date stamp. This is not dependent on the tracing level.

SNMP log

An SNMP log contains all events that are sent to SNMP traps.

Statistics log

A *statistics log* contains a summary of operational statistics for every minute that the DSA is active. When the DSA is not active, no information is written to the log, which prevents the log file from growing during inactivity. This is not dependent on the tracing level.

Summary log

A *summary log* contains a summary of every operation. This is not dependent on the tracing level or whether the DSA console is open.

Time log

A *time log* contains the time taken for each successful operation.

To configure the time log, use the *set time-log-search-threshold* command and the *set time-log-update-threshold* command.

If tracing or parsing is turned on, use the *get log* command to display the configuration of the time log.

Trace log

A *trace log* contains tracing information for all successful operations. The level of tracing written to a trace log is dependent on the level of tracing set on the DSA.

Update log

An *update log* contains detailed information for all add, modify, rename, and delete operations. This is not dependent on the tracing level. You can use the *set update-log-show-values* command to include attribute values in the update log.

Warning log

A *warning log* contains all errors and warnings, which are useful for diagnosing problems. This is not dependent on the tracing level. For descriptions of error messages, see System Messages.

Recommended Logs

You can use the following logs for DSAs in production environments:

- Alarm
- Diagnostic
- Statistics
- Summary

  Only use this for router DSAs. This file becomes very large for data DSAs.
- Trace

  When the trace level is set to none, this log acts like a console log. For test environments, set the trace level to *error*.
- Warn

Note: We recommend that you use only the logs that are actually required. Because a DSA has to write to the log files, keeping too many log files open can reduce performance.

Open a Log File

To allow a CA Directory DSA to log information, you must *open* the relevant log file. The DSA writes output to that file.

If a log file does not exist when opened, it is created. If it already exists, the DSA appends new output to the existing output in the file.

If the log file name contains the string *$S*, then the system substitutes the name of the DSA.

To open a log file, use the following command:

```
set log-type = log-name;
```

Flush a Log File

To force all buffered output to a particular log file, use the flush command.

This lets you examine a current log file offline while the normal log file is still open.

Note: Alarm logs are always flushed. This means that the alarm output is always sent to the log file immediately.

To flush a log file, use the following command:

```
flush log-type;
```

For example, to flush the summary log, enter the following command:

```
flush summary-log;
```

Close a Log File

If you close a log file, the DSA stops sending output to the log file.

When the DSA shuts down, it automatically closes any open log file.

To close a log file, use the following command:

```
close log-type;
```

For example, to close the summary log, enter the following command:

```
close summary-log;
```

List Which Logs Are Open

To inspect the names of each of the enabled log files (including the SNMP log file), use the following command:

```
get log;
```

The following is an example of the output:

```
alarm-log                = logs/democorp_alarm.log
summary-log              = logs/democorp_20070122.log
stats-log                = logs/democorp_stats_20070122.log
trace-log                = logs/democorp_trace.log
query-log                = logs/democorp_query_20070122.log
update-log               = logs/democorp_update_20070122.log
alert-log                =
cert-log        = logs/democorp_cert_20070122.log
connect-log              = logs/democorp_connect_20070122.log
snmp-log        = udp eagle port 9999
```

Historical Log Files

Most of the log files, once opened, start again each day.

For example, the following command produces a log file in the logs directory of the form *dsaname_yyyymmdd.log* for each day there is activity on the DSA:

```
set summary-log = "logs/$S.log";
```

You can use these history files to inspect auditing, accounting, billing, and statistical information.

Monitoring With DXmanager

You should use DXmanager to review the CA Directory operation periodically to ensure it continues to run smoothly. You need to monitor the directory backbone's health and load:

- **Health**—A directory's *health* is a measure of whether the directory is running smoothly. This includes whether all DSAs are running, and whether any are logging warning, error, or alarm messages.
- **Load**—A directory's *load* is a measure of the amount and type of operations being sent to that directory.

Monitoring with Traces

A DSA's *trace* is its record of almost all operations going into and out of that DSA. You can view a DSA's trace in the log files, and also by using the DSA console.

You can set the trace levels for each DSA. Each trace level records only a certain type of information about the DSA's operations.

Because the trace can be written to log files, you should not leave the trace set to a high level if it is not required. This can fill up log files and waste disk space.

We recommend that you set the trace level in the configuration files of all DSAs to a low level, such as *error*.

You can change the tracing level temporarily using the DSA console. When the DSA is next restarted, the tracing returns to the level set in the configuration file.

Protocol Tracing

Low-level protocol can be traced. The protocol trace is written to the trace log, if open. This tracing is only for debugging purposes.

To enable protocol tracing, use the following command:

```
set trace = stack;
```

Statistics Tracing

The *stats* trace option enables you to retrieve the following minute-by-minute statistical information about a DSA:

Assocs

Displays the number of bindings at the time that the statistics were collected.

NilCredit

Displays the number of times during the statistics period that the credit limit reached zero.

Queue

Displays the following numbers:

- The number of operations that have been received but are not yet processed
- The number of operations that have been sent

Together, these figures give the number of operations that this DSA is currently looking after.

For a router or a DSA performing chaining, this figure counts local requests plus all requests sent to remote DSAs to perform a local request.

MWQ

Displays the following numbers:

- The number of queued multiwrite operations that have not been sent to peer DSAs
- The total number of queued multiwrite operations

There is a discrepancy between these two numbers when multiwrite recovery in progress.

Ops

Displays the number of operations that the DSA processed in the last minute.

Entries

Displays the number of entries returned in search operations in the last minute

Change the Trace Level Temporarily

You can use a DSA console to change a DSA's trace level. This new trace level takes effect immediately. However, when you restart or reinitialize the DSA, it returns to the trace level defined in its configuration.

To change the trace level temporarily

1. Use Telnet to connect to the DSA that you want to trace.
2. Enter the following command:

   ```
   set trace = option;
   ```

3. To reset tracing to the default level, enter the following command:

   ```
   set trace = error;
   ```

   For example, to set the trace level to x500, use the following command:

   ```
   set trace = x500;
   ```

Display Trace in a Log File

You can use the log files to collect trace information.

Note: Tracing uses CPU; therefore, the DSA's response time is slower if you have set a high tracing level.

To display trace in a log file

1. Delete all of the DSA's logs.

   This means that you can be absolutely sure that you are viewing the new trace and not an old one.

2. Use the following command to open the trace log file:

   ```
   set trace = dsaname_trace.log;
   ```

3. Restart or reinitialize the DSA.

   The trace starts, outputting to the following file:

   ```
   dsaname_trace.log
   ```

Turn Tracing Off

If you turn tracing off, only alarm and fatal messages are generated.

To turn tracing off, enter the following command from the DSA console or a configuration file:

```
set trace = none;
```

Monitor the Directory Backbone Daily

To ensure that your directory continues to work well, perform the following tasks on a daily basis.

1. Ensure that the DSA is still running.

   For example, on UNIX use the commands:

   ```
   ps -ef | grep dsa
   ```

   On Windows, use the Task Manager.

2. Check the log files for each DSA. If a log file has a non-zero size, view the log and determine the cause of the error.

3. Check the disk space to ensure that there is enough free space to perform a backup of the datastores used by the DSAs.

4. Check the memory usage of each DSA.

5. (Optional) If using Windows, check the application event log for errors.

6. (Optional) If you want to monitor the load changes on a DSA, enable DSA stats tracing and check the stats log. This provides a summary of the number of operations performed each minute, and can be used to determine usage. For more information, see Using DSA Tracing (see page 263).

Monitoring Disk Space

The directory datastore is a fixed size, so you do not need to monitor its disk usage. However log files can grow, and so need increasing disk space.

Chapter 14: Manage DSAs

This section contains the following topics:

How to Stop a DSA

You can stop a DSA using DXmanager or a DSA console.

However, if a DSA has a multiwrite queue and Wait for Multiwrite is set to *True* for that DSA, the DSA does not stop. This is to prevent the queue being lost. If Wait for Multiwrite is *True*, the DSA does not stop until the queue has been flushed (that is, the target DSA has accepted all of the queued updates).

To ensure a DSA stops, we recommend that you stop processes in this order:

1. Stop any applications.

   This prevents any further updates being sent to the directory.

2. Stop any router DSA.

   This prevents updates being routed from other DSAs.

3. Stop the data DSA.

We recommend that you do not forcibly stop a DSA with queues, because this causes the queues to be permanently lost.

More information:

Prevent Multiwrite Queues Being Lost When the DSA Stops (see page 107)

Tools for Controlling DSAs

You can control DSAs using the following tools:

- DXmanager
- DXconsole
- A command line

Note: On Windows, you can use the Service Manager to start and stop DSA services.

More information:

Tools to Manage CA Directory (see page 49)

Reinitializing a DSA

If you have changed a DSA's configuration files, the DSA does not change until you restart the DSA. This is because a DSA checks its configuration only when it is starting up. When it is running, it does not refer to its configuration.

However, you can reinitialize a DSA to make it check its configuration. This lets you change a DSA's behavior without stopping and restarting it.

View DSA Current Statistics Using DX Consol

CA Directory maintains statistics about all events, services, errors, and timeouts. You can use the DSA console to monitor the DSA's current statistics.

To view DSA current statistics

1. Use Telnet to connect to the DSA you want to check.
2. Enter the following command:

   ```
   get stats;
   ```

Reset DSA current statistics

CA Directory maintains statistics about all events, services, errors, and timeouts. You can use the DSA console to monitor the DSA's current statistics.The DSA statistics are stored in counters in the DSA.

To reset a DSA's current statistics:

1. Use Telnet to connect to the DSA whose statistics you want to reset.
2. Enter the following command:

   ```
   reset stats;
   ```

View Current Operational Values of a DSA

To view the current operational values of a DSA

1. Use Telnet to connect to the DSA you want to check.
2. On the DSA console, use the following command:

   ```
   get oper;
   ```

Control DSAs by Using DXmanager

DXmanager lets you start, stop, and reinitialize DSAs.

DXmanager lets you use a single click to control any of these groups of DSAs:

- A single DSA
- All DSAs that serve a namespace partition (these DSAs all have the same prefix)
- All DSAs on a host
- All DSAs in a site
- All DSAs in a region

To control DSAs using DXmanager

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Click the Maps tab, then choose the level at which you want to control the DSAs:
   - To control all DSAs that serve a namespace partition, display the Namespace map.
   - To control a single DSA or all DSAs on a host, in a site, or in a region, display the Topology map.
3. Right-click on an item, and then select Start or Stop.

   The affected DSAs start or stop immediately.

Control DSAs by Using a Command Line

If you have direct access to a directory host, you can control DSAs on the host using a command line. You need to know the DSAs' names, but you do not need to know their port numbers.

Any of the commands that you can use in a command line can also be used in a script.

Start a DSA Using a Command Line

To start a DSA

1. Open a command line with the appropriate user permissions.

   On UNIX, use DSA user.

   On Windows, use Control User or Super User.

2. To start all DSAs, use the following command:

   ```
   dxserver start all
   ```

   To start a single DSA, use the following command:

   ```
   dxserver start dsaname
   ```

   In this command *dsaname* is the name of the directory server.

Stop a DSA Using a Command Line

To stop a DSA using the command line

1. Open a command line with the appropriate user permissions.

   On UNIX, use DSA user.

   On Windows, use Control User or Super User.

2. Enter the following command:

   ```
   dxserver stop dsaname | all
   ```

   dsaname

   Stops the named DSA only.

   all

   Stops all DSAs on this computer.

Force a DSA to Stop Using a Command Line

If you need to stop a DSA that has a multiwrite queue and W*ait for Multiwrite* is selected, use the *dxserver forcestop* command. This command stops a DSA even if it has multiwrite queues.

Note: We strongly recommend that you do not use a force stop in a production environment.

To force a DSA to stop using a command line

1. Open a command line with the appropriate user permissions.

   On UNIX, use DSA user.

   On Windows, use Control User or Super User.

2. Enter the following command:

   ```
   dxserver forcestop dsaname
   ```

Refresh a DSA Using a Command Line

If you reinitialize a DSA, it checks its configuration files and applies any changes without stopping and restarting.

To reinitialize a DSA using a command line

1. Open a command line with the appropriate user permissions.

   On UNIX, use DSA user.

   On Windows, use Control User or Super User.

2. Enter the following command:

   ```
   dxserver init dsaname | all
   ```

   dsaname

   Initializes the named DSA only.

   all

   Initializes all DSAs on this computer.

Check the Status of a DSA by Using a Command Line

To report on the status of a DSA

1. Open a command line with the appropriate user permissions.

   On UNIX, use DSA user.

   On Windows, use Control User or Super User.

2. Type the following command:

   ```
   dxserver status dsaname
   ```

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

Increase the Size of Your Datastore

You may wish to increase the size of a datastore associated with a DSA.

To extend a DSA

1. Stop the DSA.
2. Locate the dxgrid-db-size setting in the DSA configuration file.
3. Modify the dxgrid-db-size setting to define the new *increased* size you want for the datastore.
4. Run dxextenddb for the DSA.
5. Start the DSA.

Note: Data will not be lost when extending a DSA in this way.

Control DSAs by Using the DSA Console

The *DSA console* lets you connect to a DSA to give DXserver commands, receive trace information, and act as a user agent.

Stop a DSA Using the DSA Console

Before you can connect to a DSA, you must configure the DSA to accept such connections.

To stop a DSA using the DSA console

1. Use Telnet to connect to the DSA that you want to stop.
2. Enter the following command:

   ```
   shutdown;
   ```

Force a DSA to Stop Using the DSA Console

If you need to stop a DSA that has a multiwrite queue and W*ait for Multiwrite* is selected, use the *force-shutdown* command from the DSA console. This command stops a DSA even if it has multiwrite queues, and discards the items in the queues.

Note: We strongly recommend that you do not use a force stop in a production environment.

To force a DSA to stop using the DSA console

1. Use Telnet to connect to the DSA that you want to stop.
2. Enter the following command:

   ```
   force-shutdown;
   ```

   The DSA stops and any multiwrite queues are lost.

More information:

How to Stop a DSA (see page 265)
Prevent Multiwrite Queues Being Lost When the DSA Stops (see page 107)

Disconnect from a DSA by Using the DSA Console

The usual way to close the DSA console is to log out. This closes the Telnet session from the DSA:

```
dsa> logout;
```

You can also close the DSA console from the local Telnet session:

```
<control-]>
telnet> quit
```

The Telnet session closes automatically when the DSA shuts down.

Chapter 15: Manage Directory Data

This section contains the following topics:

Change Every Entry in a Directory

If you want to change every entry in a directory, create a new datastore and then load the new data from an LDIF file.

If you perform this procedure on a regular basis, you may want to set up replication between the DSAs.

To change every entry in a directory

1. Create a new datastore, using the following command:

   ```
   dxnewdb dsaname
   ```

   In this command *dsaname* is the name of the datastore that you want to empty.

2. Load the new data, using the following command:

   ```
   dxloaddb options dsaname ldif-file
   ```

   In this command *ldif-file* is the name of the LDIF file that contains the new data and *dsaname* is the name of the DSA to which you want to load the data. The options include the following:

   -O *rows*

   Specifies that DXloaddb loads operational attributes.

Example: Change Every Entry in the Democorp datastore

1. Create a new datastore using the following command:

   ```
   dxnewdb Democorp
   ```

2. Load the new data from the file *newdemocorp.ldi* using the following command:

   ```
   dxloaddb Democorp newdemocorp.ldi
   ```

Copy Data between DSAs

If you want to copy data from one DSA to another, take a copy of the datastore file.

Backing Up Data

There are two ways to back up CA Directory data. The fastest and safest way is to back up the file system. You can also export the data to an LDIF file.

Comparison of Backup Methods

The following table contrasts each backup method:

Backup Method	LDIF File	File System
Tools	**Backup:** DXdumpdb **Recovery:** DXloaddb	OS tools
Advantages	Portable between different versions of CA Directory and different vendors	Scheduled automatically as part of normal operating system procedures
Disadvantages	Slow	None
Recommended Frequency	As required to port to a different version of Directory	Nightly or incremental
Status of DSA	Running	Stopped **Note:** You can use the dump command to take a consistent snapshot of the DSA while it is running.

File System Backup

You can use a file system backup to reflect the changes that are made to CA Directory applications and program scripts.

A file system backup should include the following:

- Operating system files
- CA Directory startup scripts
- CA Directory installation files
- CA Directory configuration files
- CA Directory datastore files
- User information, including:
  - Environment (DXHOME)
  - User names and passwords

Example of File System Backup Plan

Company A is a large multinational bank that uses CA Directory as an authentication repository.

Their directory contains hundreds of millions of user entries and uses eight Solaris computers. Each computer has four data DSAs and several layers of router DSAs on it. Company A runs a 24-hours-a-day, 7-days-a-week operation with strict SLAs relating to availability, and a small weekly maintenance window.

At Company A, the directory is managed by a Directory team. The IT infrastructure is handled by a different team.

The IT team automatically backs up the file system once a day at 4 a.m.

Using an LDIF file to Back Up and Load Data

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.

CA Directory comes with the DXdumpdb tool, which lets you unload data from a datastore into an LDIF file. You can then later load the data from the LDIF file into a datastore to recover the directory content.

Back Up a Directory to an LDIF File

To back up a directory to an LDIF file

1. Log in as the user *dsa* (on UNIX) or the DXserver administrator (on Windows).
2. Use the following command to back up the datastore to the LDIF file:

   ```
   dxdumpdb -f filename -z dsaname
   ```

   -f *filename*

   Specifies the file path and name where the data is dumped.

   -z

   Specifies that DXdumpdb dumps from the copy of the datastore that is produced by the console command dump dxgrid-db.

   dsaname

   Specifies the name of the DSA.

How to Load Directory Data from an LDIF File

You can load data from an LDIF file in two ways, as follows:

- Load an entire datastore from an LDIF file, using the DXloaddb tool. For more information, see dxloaddb in the *Reference Guide*.
- Load some data into an existing datastore, using the DXmodify tool. For more information, see dxmodify in the *Reference Guide*.

Online Datastore Dump

You can take a consistent snapshot copy of the datastore of a running DSA (an online dump). The DSA completes any updates before carrying out the online dump and does not start any more updates until the copy is finished.

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

Note: Each dump overwrites the previous backup file. If you want to save the backup file, copy it to another location before the next dump.

Unique Attribute Values

Sometimes you need the attribute value in an entry to be unique. For example, for email to work each user needs a unique email ID.

You can specify which attributes must have unique values, and you can also specify a subtree within which the specified attribute must have unique values. You can specify different subtrees for different attributes.

If you implement unique attributes, ensure the client applications can handle refusals when they add or modify attribute values that already exist.

How Unique Attribute Values Work

When a client application tries to update an attribute that is set to be unique, CA Directory checks that the attribute value is not already used.

If the value is used, an error message is sent back to the client application; otherwise the client request is confirmed.

Uniqueness Checks and Access Controls

Be careful when using unique attribute values for sensitive data.

When a DSA searches entries to determine if an attribute value is unique, the search bypasses access controls. This means that a user could write a client application to determine the unique values. If these are sensitive information, this may be a security issue.

If the scope of the subtree covers more than one DSA, the first DSA (DSA-A) sends the search to another DSA (DSA-B). DSA-B obeys access controls unless DSA-A has set the trust flag trust-DSA-triggered-operations. To allow DSA-B to bypass access controls, set this flag in the DSA-A knowledge and ensure that DSA-B shares DSA-A's knowledge.

Limitation: Uniqueness Is Not Enforced in Pre-existing Data

We recommend that you enable unique attributes only on empty directories or new attributes. This ensures uniqueness.

There is no DSA mechanism to find duplicates in existing directory data. Before you enable unique attributes on a running directory, you should check for duplicate attribute values.

If you load data using the DXloaddb tool, uniqueness is not enforced.

Check for Attribute Value Uniqueness in a Subtree

You can extend checks for unique attributes to apply to all DSAs in an entire directory backbone.

Checks for uniqueness apply to any number of DSAs in a backbone as long as they are under the specified uniqueness subtree.

Attribute value uniqueness cannot be guaranteed in a distributed environment, because multiple DSAs can be updated at the same time. However this is a slim window.

To check for attribute value uniqueness in a subtree

1. (Optional) Define the subtree within which the attributes must be unique using the following command:

   ```
   set unique-attrs-subtree = DN;
   ```

2. Define the unique attributes using the following command:

   ```
   set unique-attrs = attribute [subtree = DN] [,attribute [subtree = DN]] [...] ;
   ```

   This command lets you specify which subtree each attribute should be unique within, but this is optional.

   If you do not set these subtrees here or in the *set unique-attrs-subtree* command, attributes are unique within the local prefix.

3. (Optional) Set the trust flag Trust-DSA-triggered operations in the knowledge of the DSA. Then, if the search extends to another DSA, this other DSA bypasses access controls when it performs uniqueness checking.

List the Unique Attributes in a DSA

To list the unique attributes in a DSA

1. Use Telnet to connect to the DSA that contains the unique attributes.
2. Enter the following command:

   ```
   get user;
   ```

   The output lists the DSA's configuration, including any unique attributes.

Operational Attributes

An *operational attribute* represents information used to control the operation of the directory (such as access control information), or used by the directory to represent some aspects of its operation.

If you use the all-extra-attrs or extra-attrs options, search requests include operational attributes in the reply.

Example: Read an entry returning all operational attributes

```
read-req
        entry = <c “AU”><o “Democorp”>
        all-extra-attrs;
```

The command returns all attributes of the entry, including all operational attributes, for example:

```
Entry:
        <countryName “AU”>
        <organizationName “Democorp”>
Contents:
        (objectClass organization)
        (organizationName “Democorp”)
        (telephoneNumber “(03) 9727 8900”)
        (creatorsName <countryName “AU”> <organizationName “Democorp”> <commonName
“DSA Administrator”>
        )
        (createTimestamp 19980712035245Z)
        (modifiersName  <countryName “AU”> <organizationName “Democorp”>
<organizationalUnitName “Services”> <commonName “John Smith”>
        )
        (modifyTimestamp 19980718062137Z)
        (dseType (any)3,2,16)
```

You can select particular operational attributes.

Example: Read an Entry Returning Selected Operational Attributes

```
read-req
        entry = <c “AU”><o “Democorp”><ou “Corporate”>
        extra-attrs = modifyTimestamp;
```

The command returns attributes and values of the entry, plus the requested operational attributes:

```
Entry:
        <countryname “AU”>
        <organizationName “Democorp”>
        <organizationalUnitName “Corporate”>
Contents:
        (objectClass organizationalUnit)
        (organizationalUnitName “Corporate”)
        (facsimileTelephoneNumber “(03) 9727 9722”)
        (telephoneNumber “(03) 9727 9942”)
        (modifyTimestamp 19980718062137Z)
```

Consider Not Using Operational Attributes

Operational attributes produce an overhead on all modify operations and can also decrease search operation performance. Multiwrite-DISP replication requires operational attributes, but if you do not use multiwrite-DISP replication, consider turning off operational attributes.

Prevent the Creation and Automatic Updating of Operational Attributes

By default, a DSA automatically creates, and updates operational attributes. You can prevent this behaviour.

To stop the creation and automatic updating of operational attributes

1. Add the following command to the settings configuration file:

   ```
   set op-attrs = false;
   ```

   This setting 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.

2. Re-initialize the DSA.

Sort an LDIF File

Sorting ensures the parent entries are defined before child entries, and that all entries are logically correct. This can help prevent errors when you load the file.

To sort an LDIF file, use the following command:

```
ldifsort -b bad-file infile [outfile]
```

-b *bad-file*

Writes duplicate or bad input records to the specified file.

infile

Specifies the file to be sorted.

outfile

Specifies the file to which output is to be written. If you do not use this option, the output is written to the screen or STDOUT.

Convert Data from CSV Format to LDIF

The csv2ldif tool converts CSV data into LDIF. You can then use this LDIF file to add or modify data in a directory.

For example, you could use the csv2ldif tool to convert a spreadsheet of the names and addresses of twenty new employees into LDIF, and then use the DXloaddb tool to add twenty new entries into the directory.

Counting Entries

To count the number of entries in a directory, you can search for all of the entries in the directory, and then count the number of entries returned.

However, this is not feasible in an automated system or in a system where the search returns large numbers of entries.

Instead, CA Directory provides a special entry information selector *dxEntryCount* to provide this information. This is far more efficient when result sets are large. This selector lets you count the entries within a single DSA.

CA Directory also provides the entry information selector *dxTotalEntryCount*, which counts the entries in a directory that is distributed across more than one DSA.

Example: Count Entries in a Single DSA

This example shows how to search a single DSA for entries in which the *sn* attribute starts with *C*.

This is a subtree search, which means that the filter is applied to the base object and all entries in the tree below the base object.

To count how many entries in the Democorp DSA have a surname starting with the letter C

1. Run the following search:

   ```
   dxsearch -h computer-name -p 19389 -s subtree -b "o=DEMOCORP,c=AU" (sn="C*")
   dxEntryCount
   ```

   In this search, *computer-name* is the computer that contains the Democorp DSA.

   This search returns a single entry with a *dxEntryCount* attribute, which is set to the number of entries that satisfy the search filter.

2. Check the following result for the number of entries:

   ```
   o=DEMOCORP,c=AU
   dxEntryCount=98
   ```

   This search result indicates that there are 98 entries with a surname beginning with C in the Democorp DSA.

Example: Count Entries in a Distributed Directory

In this example, the Democorp directory is distributed across several DSAs. This example shows how to count the number of entries in the whole distributed directory.

This is also a subtree search, which means that the filter is applied to the base object and all entries in the tree below the base object.

To count all of the entries in the Democorp DSA

1. Run the following search:

   ```
   dxsearch -h computer-name -p 19389 -s subtree -b "o=DEMOCORP,c=AU"
   (objectclass="*") dxTotalEntryCount
   ```

   In this search, *computer-name* is the computer that contains the Democorp DSA.

2. Check the following result for the number of entries:

   ```
   o=DEMOCORP,c=AU
   dxTotalEntryCount=1380
   ```

   This search result indicates that there are 1380 entries in the Democorp DSA.

Example: Count Entries at a Single Level in a Single DSA

This is a one-level search, which means that the filter is applied to the entries immediately below the base object, but not to the base object itself.

To count how many top-level department names in the Democorp DSA include the letter C

1. Run the following search:

   ```
   dxsearch -h computer-name -p 19389 -s one -b "o=DEMOCORP,c=AU" (ou="*c*")
   dxEntryCount
   ```

   In this search, *computer-name* is the computer that contains the Democorp DSA.

2. Check the following result for the number of entries:

   ```
   o=DEMOCORP,c=AU
   dxEntryCount=6
   ```

   This search result indicates that 6 of the top-level department names contain the letter *c*.

Example: Count the Entries at the Base Level

If you run a search designed to count the entries that satisfy a filter at the base level, the search result is either 1 or 0. This can be useful if you run the search programmatically.

The following example shows how to check whether the *organization* attribute of the Democorp base entry includes the letter "c".

To check whether the base object matches a search filter

1. Run the following search:

   ```
   dxsearch -h computer-name -p 19389 -s base -b "o=DEMOCORP,c=AU" (o="*c*")
   dxEntryCount
   ```

   In this search, *computer-name* is the computer that contains the Democorp DSA.

2. Check the following entry for the number of entries:

   ```
   o=DEMOCORP,c=AU
   dxEntryCount=1
   ```

   This search result indicates that the organization attribute for the Democorp base object does include the letter *c*.

Pattern Matching

The pattern matching mode is triggered when a filter contains the asterisk (*) or question mark (?) wildcards. In this mode, DXserver applies the pattern to find matching entries. For example, the filter, cn~=T?M J*N, would match Tim Junn, Tom Jamerson, and Tam Jinaran. This is a very precise form of matching. Use this mode if you want a lot of control over the entries that are returned.

Searching Data Using Approximate Matching

CA Directory supports Soundex-style matching as part of its approximate matching capability.

Some other directories implement approximate match (~=) exclusively as Soundex. This can be very limiting, so DXserver supports a more general approximate matching capability that has two modes—pattern matching and phonetic matching.

Phonetic Matching

Phonetic matching is the default mode. It is used when a filter contains no wildcards. It uses a Soundex-style algorithm that looks for similar sounding matches, but also permits more relaxed syllable matching and word completion.

This is a lenient form of matching. Use this mode if you prefer to get extra results rather than miss some results.

Relaxed Syllable Matching

The relaxed syllable matching mode permits non-consecutive syllables.

For example, the filter (sn ~= Stone) matches *Stein*, *Stratton*, and *Stevenson*, because the match involves *st* and *n*.

Word Completion

The word completion mode permits arbitrary suffixes. For example, the filter (cn ~= Pat) matches *Pat*, *Patrick*, *Patrice,* and *Patricia*.

This is also useful for searching by initials. For example, the filter (cn ~= a b) matches *Anne Buckley*, *Andrew Bradley*, *Andrea Bingham*, *Andy Brennan*, *Annette Bolt*, and *Anna Bromley*.

Benefits of Using Both Phonetic and Pattern Matching

Having both phonetic and pattern matching is powerful because if the phonetic matching is too lenient, then the pattern matching capability can be used to provide much more control.

For example, the filter (sn ~= ma*l) finds only surnames that begin with “Ma” and end in “l”, including Maxwell, Marshal, and Mantel.

Example: Compare CA Directory with Soundex

The following table includes some examples of searches on CA Directory compared to Soundex searches on another LDAP directory:

Filter	CA Directory Results	Soundex Results
cn ~= pete	Pete Grimm, Peter Forbes	Pete Grimm
cn ~= chris	Chris Schmith, Christoph Newport, Kirsten Vaughan	Chris Schmith, Tobias Cruse
cn ~= christopher	Christoph Newport	Christoph Newport, Kirsten Vaughan
sn ~= shel	Shelton, Sheldon	-
sn ~= shelt	Shelton	-
sn ~= sheltn	-	Shelton
sn ~= smit	Smith	-
sn ~= smith	Smith	Smith
sn ~= scmit	Schmith	-
sn ~= schmith	Schmith	Schmith
sn ~= fish	Fish, Fisher, Forbes	Fish

Referential Integrity

A directory entry has *referential integrity* if all DNs in the entry are valid, that is, point to other, existing, entries.

CA Directory provides a way to help enforce referential integrity. You can ensure that when an entry is deleted, all existing references to that entry are also deleted. This method is called *direct referential integrity,* and is equivalent to the cascade delete rule for databases. It helps keep the directory clear of *dangling references*, that is, references that point nowhere.

You can also ensure that when an entry is deleted, attributes in other entries that use a common value (rather than a DN) to identify the deleted entry are also deleted. This is called *indirect referential integrity,* and is useful if the directory uses non-DN attribute values to link different entries.

You enable referential integrity by creating one or more *referential integrity* rules in a DSA. A referential integrity rule specifies:

- The subtree of entries that triggers this referential integrity rule
- The subtree of entries in which the DSA searches for attributes that have references (direct or indirect) to the deleted entry
- The names of the attributes in which the DSA searches for references (direct or indirect) to the deleted entry

The referential integrity search begins only after the DSA has deleted the specified entry, so after an entry is deleted, there is a short time during which an entry containing a dangling reference is available to a user. Therefore, the CA Directory implementation does not guarantee referential integrity.

The CA Directory implementation of referential integrity is designed to be a tool that is used as required. It does not itself enforce referential integrity in all cases. The following constraints apply:

- There are no checks for referential integrity when an entry is created or changed.
- The DSA searches only those attribute names that are specified in the referential integrity rule, and furthermore, searches only within the subtree that is specified in the rule, so dangling references can still exist in other attributes or other subtrees.

Example: Direct Referential Integrity

Assume a directory contains an entry (for example, *Sales*) that has a *Member* attribute, where *Member* contains user DNs.

A direct referential integrity rule can ensure that when a user entry is deleted, the DSA automatically deletes that user's DN from the value of the *Member* attribute in *Sales*.

Example: Indirect Referential Integrity

Assume that each user contains an attribute *UserID,* and also that *Sales* has a *UserID* attribute, where *UserID* contains userIDs. This is similar to the previous example, with the difference that here, *UserID* contains userIDs instead of user DNs.

An indirect referential integrity rule can ensure that when a user entry is deleted, the DSA automatically deletes that user's ID from the value of the *UserID* attribute in *Sales*.

More information:

Alias Integrity (see page 394)

Create Referential Integrity Rules

You create referential integrity rules to enable referential integrity in a directory.

Define all the referential integrity rules in one place in one configuration file.

To create referential integrity rules

1. In a configuration file, enter the command to clear referential integrity rules, *clear referential-integrity,* followed by one or more instances of the command to create integrity rules, *set referential-integrity*, as follows:

   ```
   clear referential-integrity;

   set referential-integrity integrity_rule_name {

   ...

   }
   ```

2. Stop the DSA.
3. Restart the DSA.

Chapter 16: Manage Bindings

This section contains the following topics:

Bindings

When a DSA, an LDAP client, or a DUA successfully binds to a DSA, the resulting relationship is called a *binding.* Because each binding corresponds to a user, the term user can be used to mean a binding.

A DSA numbers its bindings when it creates them, starting at zero and incrementing by one.

Each binding can have a number of concurrent operations.

You should monitor the type, number, and duration of bindings to DSAs. If a DSA's performance is suffering because it has too many bindings, or because the bindings have too many operations, consider configuring the DSA to limit the type, number, and duration of bindings.

Monitoring Bindings

You can view the information that a DSA keeps about its bindings—their current state and their configuration parameters. You can also enable or disable tracing on binding activity.

Monitoring bindings lets you check the health of a DSA, and to diagnose performance issues.

View Bindings

You can view the current bindings information held by a DSA. You may want to do this to check the performance of a DSA.

To view bindings

1. Use Telnet to connect to the DSA you want to check.
2. On the DSA console, use the following command:

   ```
   get users;
   ```

   The DSA displays information about the current state of bindings.

View the Binding Configuration

You can view the binding configuration information held by a DSA.

To view the binding configuration issue the following command from the DSA console:

```
get user;
```

View the DSP Configuration

You can see the DSP configuration information. This is useful when you want to check the configuration of a DSA's DSP bindings.

To view the DSP configuration, issue the following command from the DSA console:

```
get dsp;
```

View Outgoing Bindings

You can view the binding information for outgoing bindings. This is useful if you want to see DSP bindings to other DSAs.

To view outgoing bindings, issue the following command from the DSA console:

```
get online-dsas;
```

Control Tracing of Bindings

You can enable or disable traces of a binding's operations. Traces are useful for diagnosing performance issues with bindings.

To control tracing of bindings

1. Use Telnet to connect to the DSA you want to check.
2. (Optional) Use the following command to change the trace level:

   ```
   set trace = none | trace-level;
   ```

3. Use the following command to determine the number of the binding that you want to trace:

   ```
   get users;
   ```

4. Use the following command to enable tracing of the binding:

   ```
   trace enable assoc = binding_number;
   ```

5. When you are finished, use the following command to stop tracing the binding:

   ```
   trace disable assoc = binding_number;
   ```

Example: Trace a Binding at Error Level

This example shows you how to show the error messages that are produced from an binding.

1. Set the tracing to x500 level:

   ```
   set trace = x500;
   ```

2. Determine the binding number of the user that requires tracing:

   ```
   get users;
   ```

3. Use the binding number in the trace enable command:

   ```
   trace enable assoc = binding_number;
   ```

Closing Bindings

You can close a binding by closing the TCP link to a DSA. You can also explicitly close bindings by using the abort command.

You can also close outgoing bindings from a DSA to one or all other DSAs by using the unbind command.

Whichever of these methods is used, the DSA closes the binding immediately, discarding any current operations on the binding.

Abort All Bindings

You normally only abort all bindings as a prelude to stopping the DSA. In this case you may want to disable new bindings to the DSA before you abort all bindings.

To abort all bindings

1. Use Telnet to connect to the DSA on which you want to abort all bindings.
2. Use the following command:

```
abort all-users;
```

Abort One Binding

Aborting a binding is a standard way to disconnect a user from a DSA. You may want to explicitly abort a binding to allow the DSA to accept new bindings, for example if the DSA has reached its maximum number of allowed bindings.

To abort one binding

1. Use Telnet to connect to the DSA on which you want to abort the binding.
2. (Optional) List all bindings, using the following command:

```
get users;
```

3. Stop the binding using the following command:

```
abort user number;
```

number

Specifies the binding that you want to abort.

Close Outgoing Bindings

You can close outgoing bindings from a DSA to another DSA, or to all other DSAs.

To close outgoing bindings

1. Use Telnet to connect to the DSA on which you want to abort the bindings.
2. (Optional) On a DSA console, find the number of remote DSAs by using the following command:

   ```
   get dsas;
   ```

3. Stop the outgoing binding using the following command:

   ```
   unbind all | dsa dsa-num;
   ```

   dsa-num

   Specifies the remote DSA. The command aborts all outgoing bindings from the local DSA to the DSA that is specified by *dsa-num*.

Disable New Bindings

You can tell the DSA to refuse any requests for a new binding. You may want to do this, for example, as a prelude to gracefully closing a DSA.

To disable new bindings

1. Use Telnet to connect to the DSA for which you want to prevent additional bindings.
2. On a DSA console, use the following command:

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

   Note: If a user tries to bind to the DSA after you have issued this command, it refuses and returns an error message.

3. (Optional). You can re-enable new bindings with the following command:

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

Setting Binding Limits

To ensure the good performance of a DSA you should set limits on the bindings.

Limit the Number of Bindings

You can limit the number of concurrent bindings that a DSA supports.

If a DSA receives a request for a new binding when it has already reached the maximum number of bindings, then it looks for a binding that it can abort to make room for the new request. It aborts an existing binding if the binding has reached its maximum bind time or its maximum guaranteed idle time. If there are no such bindings then the DSA rejects the request for a new binding.

To limit the number of bindings

1. Use Telnet to connect to the DSA.
2. On the DSA console, use the following command:

   ```
   set max-users = max-bindings;
   ```

   max-bindings

   Defines the maximum number of concurrent bindings.

Example: Limit Bindings to 100

The following command limits the number of concurrent bindings to 100:

```
set max-users = 100;
```

Example: Remove the DSA Limit to Bindings

The following command removes any DSA limit to the number of bindings. However, the operating system sets an upper limit to the number of bindings per DSA, which is based on the maximum number of file descriptors per process.

```
set max-users = 0;
```

Limit the Number of Operations per Binding

You can limit the number of concurrent operations per binding.

You can also limit the total number of concurrent operations for a DSA. To do that, see Limit the Number of Concurrent Operations (see page 310).

To limit the number of operations per binding, use the following command from a DSA console:

```
set credits = max-operations;
```

max-operations

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

Set a Maximum Idle Time for DSP Bindings

A sending DSA can close a binding to a peer DSA if the peer DSA does not respond quickly enough. The time allowed is called the maximum DSP idle time.

You can set the maximum DSP idle time in the following places:

Backbone

If you set the value here then it applies as a default value for the whole directory backbone.

Each namespace partition

If you set a value here it overrides the backbone default value, if one is set.

To set a maximum idle time for DSP bindings

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab, and then choose the place you want to set the maximum DSP idle time.
   - To set the backbone default, display the Topology map.
   - To set the value for a namespace partition, display the Namespace map.
4. Right-click on the item that you want to change, and then select Edit properties.
5. Find the maximum DSP idle time setting (in the bindings area), and then enter the value.
6. Click OK or Finish.

Set the Maximum Guaranteed Idle Time

You can set a maximum guaranteed idle time. If the DSA already has the maximum number of bindings then it must abort one of these before it can accept a new one. Setting a maximum guaranteed idle time allows the DSA to abort an existing binding that has been idle for a long time, and so increases the availability of the directory service while still limiting the number of concurrent bindings.

To set the maximum guaranteed idle time, issue the following command from a DSA console:

```
set user-idle-time = num-secs;
```

num-secs

Defines the number of seconds that a binding can be idle without risking being closed.

Set the Maximum Binding Time

You can ensure that a DSA aborts any binding that lasts longer than a specified time. This ensures that bindings do not attach to the DSA indefinitely.

To set the maximum binding time, use the following command from a DSA console:

```
set max-bind-time = num-secs;
```

num-secs

Defines the maximum duration for any binding in seconds.

Note: If *num-secs* is zero, the DSA does not set any maximum binding time.

Process Concurrent Binds from CA SiteMinder

CA SiteMinder uses a single thread for authentication. If CA SiteMinder does not detect a Netscape, iPlanet, or SunOne server, this thread serializes authentication binds to the directory, which slows performance.

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

- Pretend to be a Netscape server to SiteMinder (see set mimic-netscape-for-siteminder Command)
- Process concurrent binds (see set concurrent-bind-user Command)

Chapter 17: Manage Operations

This section contains the following topics:

Types of Operations

CA Directory is an X.500 directory that also uses LDAP. This means that CA Directory can deal with any LDAP or X.500 query which is also called a query.

Operations divide naturally into three categories as follows.

Operation	Type of Operation	X.500 or LDAP	Description
search	Search Query	LDAP and X.500	Searches for entries that match the filter
compare	Search Query	LDAP and X.500	Compares a supplied value against the values stored in an entry (often used for passwords)
read	Search Query	X.500	Extracts information from an entry
list	Search Query	X.500	Lists the entries immediately below a given entry
abandon	Search Query	LDAP and X.500	Abandons an operation (search type operations only)
add	Update Query	LDAP and X.500	Adds a leaf entry
remove	Update Query	LDAP and X.500	Deletes a leaf entry
modify	Update Query	LDAP and X.500	Adds or removes attributes or attribute values (not distinguished values) of any entry
modify DN	Update Query	LDAP and X.500	Changes the name of any entry
bind	Binding	LDAP and X.500	Creates a binding to a DSA
unbind	Binding	LDAP and X.500	Releases a binding to a DSA

Searches Are Faster Than Updates

Directories are optimized to provide excellent performance for search queries, which only read from the database. Update queries are slower because the changes have to be written to the disk.

Take this into account when designing your directory's performance.

Simple and Complex Searches

The following table lists the types of searches that are simple:

Simple Search Types	Example Filter
Exact match	cn=john smith
Initial substring	cn=john s*
Simple searches combined with ANDs or ORs	(&(cn=john smith)(cn=john s*)) (|(cn=jon*)(cn=john*))
Base-object searches, regardless of the filter	(!(cn=john smith))

The following table lists the types of searches that are complex:

Complex Search Types	Example Filter
Final substring	cn=*th
Present	(objectclass=*)
Sounds like	cn~=john smith
Not	(!(cn=john smith))
Complex searches combined with ANDs or ORs	(&(objectclass=*)(cn~=john smith))
Greater-than, less-than, greater-than or equals, less-than or equals	sn<=b
One-level and subtree searches, regardless of the filter	

DSA Handling of Multiple Concurrent Queries

In CA Directory each DSA can process multiple queries concurrently. This means that slow queries, such as updates, do not delay quick ones such as simple searches.

CA Directory enforces limits on the number of queries that the DSA concurrently processes. Use the set user-threads command to specify the number of concurrent user threads per DSA. The performance of a user thread does not vary significantly for different types of query.

More information:

Types of Operations (see page 299)

Abandon an Operation

The result returned by an abandoned operation depends on how far the operation progressed before being abandoned. The display shows either partial results (with the partial outcome qualifier set to the appropriate problem) or the following error message:

```
Service error: administration limit exceeded
```

An update operation cannot be abandoned.

To abandon an operation

1. Use Telnet to connect to the DSA whose operation you want to abandon.
2. List the current binding numbers, using the following command:

   ```
   get users;
   ```

3. Identify the binding number of the operation that you want to abandon.
4. Abandon the operation, using the *abandon* command:

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

Limit Operations

You can set the following limits on some directory operations:

Size

Limits the number of entries that the operation can return.

You can set a size limit on any search or list operation.

Time

Limits the time that the operation can run.

You can set a time limit on any inquire operation (search, compare, read, list, and abandon).

If an operation reaches one of these limits, the operation is not abandoned. Instead, the following happens:

1. The operation is stopped.
2. Any partial results are returned.
3. The following error is also returned:

```
Partial Outcome Qualifier:
Limit Problem: Admin limit exceeded
```

More information:

Setting Binding Limits (see page 295)

Methods for Setting Operation Limits

The following table summarizes the advantages and disadvantages of each method of setting limits on operations:

Method	Description	Advantages	Disadvantages
DSA-based limit (see page 305)	Limits can be set for an entire DSA. All users binding to the DSA are constrained by these limits, unless they have role-based limits.	Simple to configure	Must stop and start DSA for changes to take effect Cannot set different limits for administrative accounts or applications
Role-based limit (see page 306)	Limits can be assigned to a role. When a member of that role binds to the directory, the limits for that role apply to the user.	Set limits once for a role, and all users with that role inherit the limits Can change the limits without having to restart the DSA Can run batch services that may return large search results without the need to run a separate DSA	-
Operational Limit (see page 309)	Limits can be included in a search request. These limits apply to that search request only.	Applications can control the amount of data returned	Relies on applications to set appropriate limits for their searches

How the Methods for Setting Limits Interact

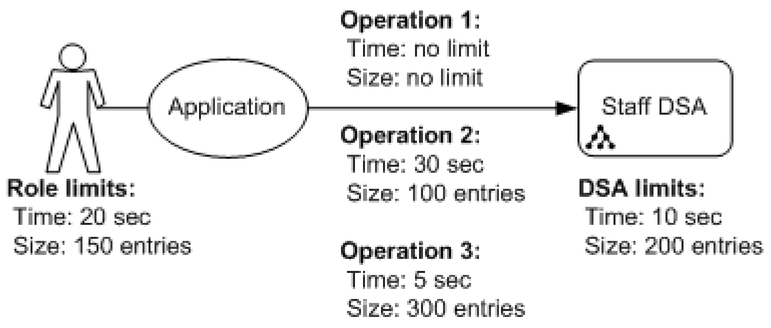
You can combine the methods for setting limits on operations. The following rules apply:

- A role-based limit takes precedence over a DSA-based limit.
- An operational limit that is less restrictive than a DSA-based limit takes precedence over that DSA-based limit.
- An operational limit that is less restrictive than a role-based limit takes precedence over that role-based limit.
- If the user has a role and limits defined in the operation are larger than the limits configured for the role, then the role's limits are enforced.

Example: Limits Set in Different Places

The example directory system shown in the diagram below has limits set in three places:

- The user is a member of a role that has limits
- The operations each have limits
- The DSA has limits



The following table shows the actual limits that apply for each operation run by this user:

Operation	Limits
Operation 1	Time: 20 sec Size: 150 entries
Operation 2	Time: 20 sec Size: 100 entries
Operation 3	Time: 5 sec Size: 150 entries

If another user, who is not a member of any roles, runs these operations, the following limits apply:

Operation	**Limits**
Operation 1	Time: 10 sec Size: 200 entries
Operation 2	Time: 10 sec Size: 100 entries
Operation 3	Time: 5 sec Size: 200 entries

DSA-Based Limits

You can limit the time and size of queries for an entire DSA.

If you also set limits for a request or a role, the largest setting applies.

Limit the Time for Operations

You can restrict the maximum time that the list and search services can run. If an operation exceeds this limit, the following error is returned:

```
Administration limit exceeded
```

To limit the number of concurrent operations on a DSA

1. Open the *limits* configuration file in a text editor.
2. Add the following command:

   ```
   set max-op-time = seconds;
   ```

3. Save the file, and then stop and start the DSA.

Limit the Number of Entries Returned

You can restrict the maximum number of entries that can be returned that any search or list operation on a DSA. If an operation exceeds this limit, the following error is returned:

```
Administration limit exceeded
```

To limit the number of concurrent operations on a DSA

1. Open the *limits* configuration file in a text editor.
2. Add the following command:

   ```
   set max-op-size = number-entries;
   ```

3. Save the file, and then stop and start the DSA.

Role-Based Limits

If roles are set up, you can apply operational limits to a role. All members of that role inherit the limits you have set. This works for both static and dynamic roles.

Role-based limits are applied to each connection, which means that any changes to these limits do not take effect until the users reconnect.

Because role-based configuration is set using attributes, you do not need to restart the DSA for the change to take effect.

More information:

Groups and Roles (see page 207)

How Role-Based Limits Work

If role-based limits are set up, the following happens when a user binds to the directory:

1. The user binds to the DSA.
2. During the bind, the DSA checks whether the user is a member of a role.
3. If the user is a member of a role that has limits, these limits are stored against the active connection.

   If a user has multiple roles, each with limits, then the largest values are used.

   If no role-based limits apply to a user, then the limits in the DSA apply.

4. Every search performed on the connection uses these limits.

Apply Operational Limits to a Role

You can set different operational limits for each role. Because role-based limits are set using attributes, you do not need to restart the DSA for the change to take effect.

To apply operational limits to a role

1. Set up a static or dynamic role in the directory (see page 217).
2. Add the object class *dxRoleBasedConfig* to the role entry.
3. Set one or both of the following limits:
   - Add the attribute *dxSizeLimit* to the role entry and change the value to the maximum number of entries that can be returned by a search.
   - Add the attribute *dxTimeLimit* to the role entry and change the value to the time (in seconds) before the operation is canceled.
4. Save the changes to the role entry.

   The new operational limits apply to users with that role when they next connect to the directory.

More information:

Setting Up Static Directory Roles (see page 219)
Setting Up Dynamic Directory Roles (see page 223)

Example: Add Limits to an Existing Role

This example shows how to use JXweb to add new limits to an existing role in the Democorp directory.

This example assumes that the dynamic role *Engineer* is already set up.

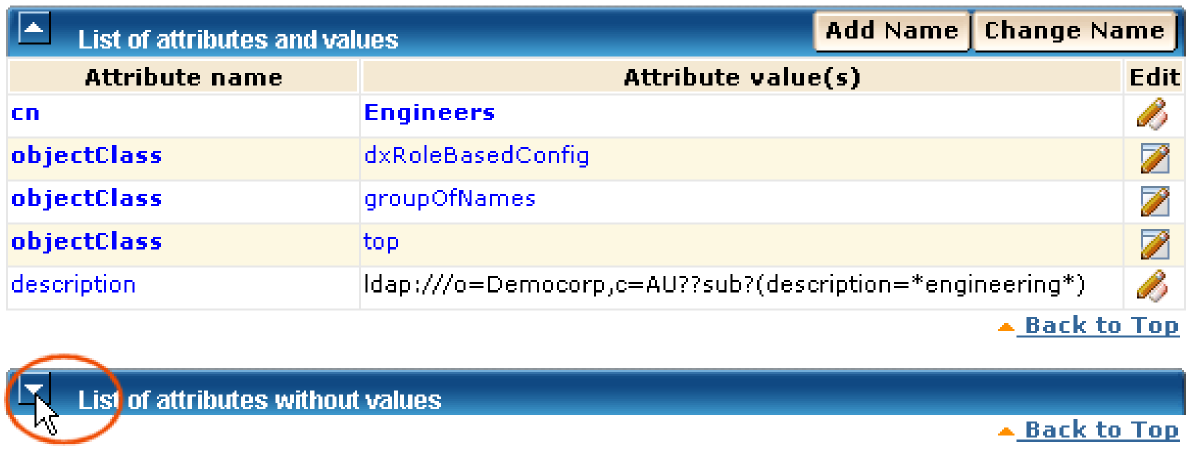
To add limits to the dynamic role *Engineer*

1. Open JXweb and connect to the Democorp directory.
2. Navigate to the Engineer entry, and select it.

   The Engineers entry appears in the right pane.
3. Click the Edit icon next to either of the object classes.

   The Add or Remove Object Classes page appears.
4. Scroll down the Available list to find the *dxRoleBasedConfig* object class.
5. Select this class and then use the arrow to add it to the entry.
6. Click Submit, and click OK in the message box that appears.

   The entry now includes the *dxRoleBasedConfig* object class. You now need to add values to one or both of the *dxSizeLimit* and *dxTimeLimit* attributes.
7. Display the attributes without values by clicking the arrow beside *List of attributes without values*, as shown in the following screenshot:


List of attributes and values
Add Name
Change Name
Back to Top
List of attributes without values
Back to Top

Attribute name	Attribute value(s)	Edit
cn	Engineers	
objectClass	dxRoleBasedConfig	
objectClass	groupOfNames	
objectClass	top	
description	ldap:///o=Democorp,c=AU??sub?(description=*engineering*)	

8. Set limits for this role by doing one of the following:

   ■ Click the Edit icon next to the *dxSizeLimit* attribute, enter the number of entries, and then click Modify.

   ■ Click the Edit icon next to the *dxTimeLimit* attribute, enter a time in seconds, and then click Modify.
9. To set a value for the other attribute, repeat Steps 5-8.

Operational Limits

When you submit a search or list request to the directory, you can specify a maximum time limit and maximum number of entries to be returned.

If the limits defined in the operation are larger than the limits configured in the DSA, then the DSA's limits are enforced.

If the user has a role and limits defined in the operation are larger than the limits configured for the role, then the role's limits are enforced.

Example: Send a Limited Search to a DSA with Limits in Its Configuration

In this example, you send a search request to a DSA that also has limits set in its configuration.

The DSA has the following limits set:

```
set max-op-time = 60;
set max-op-size = 100;
```

You want to search this DSA for all entries below a base DN that have the *sn* attribute populated. You want to limit the search to 1000 seconds and a maximum of 10,000 entries to be returned.

1. Use the DXsearch tool to submit the following request to the DSA, specifying size and time limits:

   ```
   dxsearch -s sub -b base-DN -l 1000 -z 10000 -h hostname -p port-number "(sn=*)"
   ```

   If you try this yourself, ensure that you replace the values for the *-b*, *-h*, and *-p* options.

2. The DSA processes the search.

3. When 100 entries have been found, the DSA stops the operations, and returns the following error to the LDAP client application:

   ```
   Partial Outcome Qualifier:
   Limit Problem: Admin limit exceeded
   ```

Limit the Number of Concurrent Operations

You can restrict the maximum number of DSA operations in progress at the same time. This number includes operations that the DSA has waiting for completion in an internal queue, and is usually a lot bigger than the number of user threads.

To limit the number of concurrent operations on a DSA

1. Open the limits configuration file in a text editor.
2. Add the following command:

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

3. Save the file, and then stop and start the DSA.

Search Profiles

Search profiles give an administrator a way to restrict the searches that specified users can perform. This is useful, for example, if you want to ensure that some users do not perform potentially performance-affecting searches.

A search profile specifies a category of searches. It defines the category in terms of the scope and the filter of the search. For more information on how to specify a search profile, see the *set allow-search* command.

When a user requests a search, the DSA checks the search request against the user's search profiles and only accepts the request if the search matches one of the profiles.

Example: A Restrictive Search Profile

A simple example is a search profile that only allows searches that have a base-object scope and an equality match filter. If a user is restricted to such a search profile, then neither of the following LDAP searches would be allowed:

```
ldapsearch -h host:30000 -b "cn=SIM,ou=Views,o=ACME" -s subtree (cn=John Smith)
SIMCardNumber
ldapsearch -h host:30000 -b "cn=SIM,ou=Views,o=ACME" -s baseobject (baseObject=*)
SIMCArdNumber
```

However, the following LDAP search would be allowed:

```
ldapsearch -h host:30000 -b "cn=SIM,ou=Views,o=ACME" -s baseobject (cn=John Smith)
SIMCArdNumber
```

Example: A Non-restrictive Search Profile

You can specify a search profile that allows any search. If this profile is assigned to a role, then any user with that role can perform any search, even if the user's other roles have more restricted search profiles. This is useful if you want to create a group of unrestricted users, while still restricting the searches of users who are not in the group.

Using Search Profiles

Once you have defined a search profile, you can make it the default search profile, or you can assign it to one or more roles.

If you make a search profile the default profile, then it applies to all users who do not have a role with a search profile.

If you assign the search profile to a role, then when a user is assigned to that role, the user is restricted to searches allowed by that profile. If a user is assigned to multiple roles, and so has multiple search profiles, then that user can perform a search if any of the search profiles allows the search.

Each role can have at most one search profile. If a role does not specify a search profile, then the role plays no part in allowing or restricting searches by search profiles.

If a user request has been allowed after being tested against a search profile, then the DSA ignores the DXmanager configuration items for *limit list* and *limit search* when processing that request.

Use the following commands to manage search profiles:

- *set allow-search*. Creates a search profile.
- *get allow-search*. Displays a list of all search profiles.
- *clear allow-search.* Deletes all search profiles.
- *set allow-search-default*. Makes a search profile the default.

For information on how to assign a search profile to a role, see Assign a Search Profile to a Role (see page 313).

Create Search Profiles

Create search profiles if you want to restrict the searches that specified users can perform.

Define all the search profiles in one place in one configuration file.

To create a search profile

1. In a configuration file, enter the command to clear search profiles, *clear allow-search*, followed by one or more instances of the command to create search profiles, *set allow-search,* as follows:

   ```
   clear allow-search;
   set allow-search profile_name ={
   ...
   ```

2. Stop the DSA.
3. Restart the DSA.

If a role has the attribute *dxAllowSearch* and the attribute value equals a defined search profile name, and then any user in that role is now restricted to the searches allowed in the search profile.

Assign a Search Profile to a Role

If you assign a search profile to a role, then you can restrict the searches that users assigned to the role can perform. If the user does not have any role with a search profile, then the user is subject to the default search profile (if one is defined).

To assign a search profile to a role

1. Add the *dxRoleBasedConfig* auxiliary object class to the group.
2. Add the attribute *dxAllowSearch* to the role, and for the attribute value give a search profile name.

   Any user assigned to that role is now restricted to searches allowed by that search profile unless the search is allowed by another search profile that also applies to that user.

Create a Default Search Profile

Create a default search profile if you want to restrict users' search requests by default. To give a user other search profiles, assign that user to roles with other search profiles.

To create a default search profile

1. Create a search profile by using the *set allow-search* command.
2. Make the search profile the default one by using the *set allow-search-default* command.

Reject Operations

You can set a DSA to reject the following operations:

- Update requests
- List requests
- Complex search requests

Shadow DSAs

A router does not direct update requests to DSAs that are marked *Read-only* or *Shadow*.

The Read-only setting prevents all updates to the directory data, including DISP and multiwrite updates.

The Shadow setting prevents all updates to the directory data, except through DISP or multiwrite. Clients can query, but cannot update, the DSA using DAP or LDAP.

Make a Namespace Read-only

If a namespace is read-only then all DSAs that serve that namespace reject all update requests, and no router DSAs sends any update requests to those DSAs.

This guarantees update security and overrides any access controls.

To make a namespace read-only

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Namespace map.
4. Right-click the namespace partition that you want to make read-only, and then select *Edit properties*.
5. In the *Namespace partition* area, select the *Read only* check box, and then click OK.

   The configuration for this namespace partition has been updated.
6. To apply the changes to the DSAs that serve this namespace partition, deploy the new configuration.

Reject Some Types of Requests

You can prevent a DSA from accepting the following types of requests:

List requests

This is useful if the DIT has a very flat structure and there are thousands of entries under one node. In this case, the list operation is not useful.

Complex search requests

You can prevent a DSA from accepting searches that have no filter or have a complex search filter.

This is useful if a DSA is set up to handle large numbers of simple lookup searches, and a high throughput is very important. Large complex searches can put unwanted pressure on the performance of the DSA.

To reject some types of requests

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Ensure that DXmanager is in Edit mode.
3. Click the Maps tab and display the Topology map.
4. Navigate to the DSA that you want to change, right-click on it, and then select *Edit properties*.
5. In the *Performance* area, select the types of request that the DSA should not accept, and then click OK.

   The configuration for this namespace partition has been updated.
6. To apply the changes to the DSAs that serve this namespace partition, deploy the new configuration.

Note: If you want to prevent different types of requests depending on the user sending the request, use search profiles. Search profiles let you allow or disallow different requests for different users. For more information, see Search Profiles (see page 311).

More information:

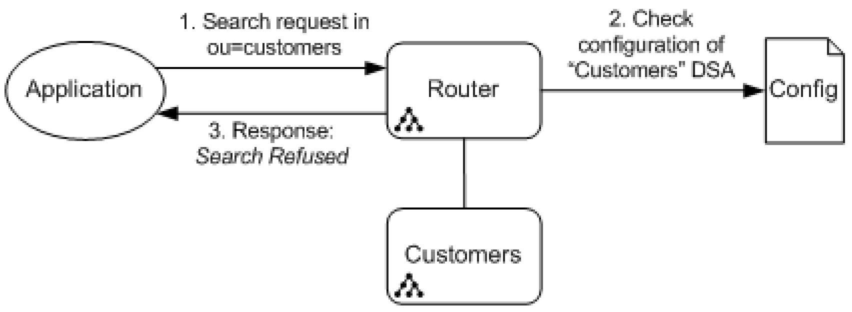
Simple and Complex Searches (see page 300)

How the Limit Search Setting Works

DXmanager lets you limit the kinds of requests that a DSA can respond to.

Because the configuration is shared, any DSA that needs to route a query to another DSA can first check whether that query would be rejected.

The following diagram shows how the configuration for a *Limit Search* DSA works:


1. Search request in ou=customers
2. Check configuration of "Customers" DSA
3. Response: *Search Refused*
Application
Router
Config
Customers

1. The router DSA receives a complex search request from a client application. The router DSA cannot fulfill the search request, but it knows that the Customer's DSA can.
2. The router DSA checks the configuration for the Customer's DSA to see whether there are any conditions for searching the Customer's DSA.
   The *Limit Search* item indicates that the Customer's DSA rejects any complex searches.
3. The router DSA returns a Search Refused response to the client.

Chapter 18: Manage User Accounts and Passwords

This section contains the following topics:

User Accounts

A user account is a directory entry with a user password. This password must be stored in the attribute *userPassword*.

You can use CA Directory to manage these user accounts, including locking and unlocking user accounts, setting up password quality rules, and assigning account suspension rules.

You can also create groups and roles in the directory. While roles are usually only useful for managing user entries, you can use groups to manage other kinds of entries.

User Account States

A user account can be in one of the following states:

Active

The user can log in.

Expired

The user cannot log in because they have not changed their password recently.

Suspended

The user cannot log in because they have tried to log in with invalid credentials too many times, or they have not logged in recently.

Locked

The user cannot log in because an administrator has locked the account.

Password Policies

A policy is a set of rules that defines behavior.

You can use CA Directory to set up password policies to manage your user accounts. A password policy applies to the *userPassword* attribute, which contains the password that users enter when binding to a DSA using simple authentication.

Only one password policy can be active within a DSA.

The password policy settings control the following areas:

- Authentication
- Password strength
- Account control and password management

You can define password policies in the directory and in an application. However, we recommend that you do not set up a password policy within CA Directory if another application also handles password policies.

A password policy is defined by the *set password* commands, which are listed in Commands for Managing User Accounts in the *Reference Guide*.

Password Security

Passwords are encrypted before they are stored in the directory. This prevents the possibility of interrogating the directory directly for the value of any passwords.

For more information about encryption algorithms, see Password Storage (see page 151).

Passwords are always single-valued. This means that an administrator cannot add a secret value and use this as an unpublished entry point into the DSA.

Password logins are secure only if each person can change only his or her own password. Ensure that you set access controls so that each user can update only their own password. Also, ensure that administrators are allowed to update any user password.

Password Commands Requiring an LDAP Client

Some password commands can only be used with LDAP clients that are aware of LDAP password policy controls (for example, LDUA and the PAM-LDAP client).

The following commands help you enhance account control:

- set password-force-change command
- set password-age-warning-period command

CA Directory uses the following command to mimic the nonstandard functionality of some other directories:

- set password-mimic-netscape-response-controls command

This section of the CA Directory password policy is specified in an Internet Draft on the IETF home page. The specification of its operation can change over time. Also, the name of the draft document changes as revisions are made. At the time of writing, the document name is *draft-behera-ldap-password-policy-09.txt*.

```
PasswordPolicyResponseValue ::= SEQUENCE {
    warning [0] CHOICE {
    timeBeforeExpiration    [0] INTEGER (0 .. maxInt),
    graceLoginsRemaining    [1] INTEGER (0 .. maxInt) } OPTIONAL,

    error   [1] ENUMERATED {
    passwordExpired                 (0),
    accountLocked                   (1),
    changeAfterReset                (2),
    passwordModNotAllowed           (3),
    mustSupplyOldPassword           (4), <== Not required (handled by bind)
    insufficientPasswordQuality     (5),
    passwordTooShort                (6),
    passwordTooYoung                (7),
    passwordInHistory               (8) } OPTIONAL }
```

timeBeforeExpiration and *graceLoginsRemaining* are provided where appropriate. For example, password policy must be enabled in CA Directory.

Plan a Password Policy

Before you start setting up password rules, plan your global password policy.

As with access controls, you should write your password policy out carefully in English, and then create the password rules to implement the policy.

We recommend that you use the following principles while you are developing your password policy:

- Plan your password policy
- Keep it simple
- Design from application requirements

How to Set Up a Password Policy

You should apply any new or changed password policy to a test directory first.

To design and create a password policy, follow these steps:

1. Plan your password policy, including the following:
   - The minimum quality of passwords
   - The lifespan of passwords
   - How you want to administer accounts, including account locking
2. Set up a test directory that contains some typical user account entries.

   Each user account is actually an entry in the directory that contains a *userPassword* attribute.
3. Create the password rules to implement your policy, including the following:
   - Rules that enforce password quality
   - Rules that enforce password lifespan
   - Rules to help you administer the user accounts in the directory
4. Enable password management in the test directory.
5. Test and fix the password policy.
6. When you are sure that your password policy works correctly, create the same password rules in your live directory.
7. Test the new password policy in your live directory.

Multiple Password Policies

Each DSA can have only a single set of password rules. You cannot apply different password policies to users stored within the same DSA.

However, you can split the users into separate subtrees, and have each subtree serviced by a separate DSA. This lets you set up a different password policy for each group of users.

Create a Password Policy

Once you have planned your password policy, you need to configure the appropriate settings.

Each policy rule is actually a command that you should include in a .dxc file in the DXHOME/config/settings folder.

You can create rules to enforce the following:

- Password quality (see page 323)
- Account expiry (see page 328)
- Account suspension (see page 331)

More information:

Example: Simple Password Policy (see page 335)
Example: Complex Password Policy (see page 336)

How to Configure Password Quality Rules

You can set up rules to ensure that users choose only strong passwords.

If you have set rules about password quality, these rules are applied when users try to change their own passwords. If the new password does not pass the tests, the users have to try again with another new password.

When an administrator changes a password for a user, the password policy rules are not usually applied. The rules are applied the next time that the user changes his or her password. However, you can force the password quality rules to be applied all of the time, even when an administrator resets a user's password. For more information, see Force Users to Change Passwords after Reactivation (see page 341).

Set Password Length

To set the length of new passwords, use one or both of these commands:

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

Set Minimum Numbers of Types of Characters

CA Directory includes commands that let you specify the minimum numbers of various types of characters that each new password must contain.

To set the minimum number of particular types or characters, use one or more of these commands:

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

Example: Specify Characters in Passwords

In this example, you want to force users to create passwords that include the following characters:

- At least two numbers
- At least one non-alphanumeric character
- At least one uppercase character
- At least one lowercase character

To create this policy, use the following commands:

```
set password-policy = true;
set password-numeric    = 2;
set password-non-alpha-num = 1;
set password-lowercase   = 1;
set password-uppercase   = 1;
```

Users cannot create the following passwords:

- *abcd12#*

  This password contains no uppercase characters.

- *ABCD!@#$*

  This password contains no lowercase characters and no numeric characters.

They can create the following passwords:

- *Abcd12#*
- *ABCd1234!@#$*

Limit Repetition of Characters

To limit repetition within the password, use the following command:

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

This lets you set the number of times a character can be repeated within a password.

Example: Prevent Character Repetitions

You have set up the following password policy in the Democorp DSA:

```
set password-policy = true;
set password-max-repetition = 2;
```

Users in this directory cannot create the following passwords:

- helllo
- lillly

However, they can create the following passwords:

- hello
- llily

Limit Repetition of Substrings

You may want to prevent users from choosing passwords that consists of repetitions of the same strings, such as *asdasdasd*.

To prevent repetition of a substring within a password

1. Set the minimum length of the substring that you want to check for, using the following command:

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

2. Set the number of times a substring can be repeated, using the following command:

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

Example: Limiting Substring Repetition

You have set up the following password policy:

```
set password-policy = true;
set password-max-substring-repetition = 1;
set password-min-length-repeated-substring = 3;
```

Users in this directory *cannot* create the following passwords:

- moomoo
- mooomooo

However, they *can* create the following passwords:

- momo
- mooomouu

Prevent the User Name from Appearing in the Password

To prevent users from including their own names in the password, use the following command:

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

The password cannot be a substring of the user's name and the user's name cannot be a substring of the password.

The user's name is taken to be the last RDN in their DN.

Example: Prevent Usernames in Passwords

You have set up the following password policy in the Democorp DSA:

```
set password-policy = true;
set password-username-substring = true;
```

The user with the DN <c AU><o DEMOCORP><ou Corporate><ou Administration><cn "Craig LINK"> has the name *Craig LINK*.

This user cannot create the following passwords:

- *craig*
- *link*
- *clink*
- *23craig4*

Prevent User Details from Appearing in the Password

To prevent users from including their own details in their passwords, use the following command:

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

This lists the attributes that contain details about the user that you do not want to be used in new passwords. The password cannot be a substring of the substring attrs and the substring attrs cannot be a substring of the password.

The user's details are taken from the values of the attributes in their own entry.

We recommend that you include only attributes that have string syntaxes. Values of other syntaxes may not be picked up by the password substring check.

Example: Prevent User Details in Passwords

You have set up the following password policy in the Democorp DSA:

```
set password-policy = true;
set password-substring-attr = title;
```

The entry for Craig Link includes the following attributes and values:

Attribute	Value
cn	Craig LINK
description	Railways

Attribute	Value
title	Communications Engineer

Craig Link cannot create the following passwords because they are substrings of the *title* attribute value:

- *engineer*
- *Communications Eng*

He can create the following passwords:

- *railways*

  This is acceptable because the *description* attribute is not listed in the *set password-substring-attr* command.

- *CommunicationsEngineer*

  The space is missing, so this is not a substring of *Communications Engineer*.

Prevent Users from Reusing Passwords

To prevent users from reusing passwords, set the maximum number of passwords to retain in the history, using the following command:

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

How to Set Account Expiry Rules

Use password policy settings to control when user accounts move from active to expired.

An account expires when the password has not been modified for the configured period.

Set Accounts with an Old Password to Expire

Use the *set password-age* command to set an account expiry time. An account expires when the number of days after the last password update reaches the value set with the *set password-age* command.

To set the number of days for which a password is valid, use the following command:

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

Example: Force Users to Create a New Password Every Four Days

You want to make users who work with sensitive information change their passwords frequently.

To do this, use the following commands:

```
set password-policy = true;
set password-age = 4;
```

Now, each user must change his or her password before it is four days old.

More information:

Reactivate a Suspended or Expired Account (see page 340)

Set the Minimum Life Span of Passwords

You can set the minimum life span of a password. This is the number of days since a password was changed until it can be changed again.

You can use this to prevent people from changing their password many times to fill up the password history.

To set the minimum life span of passwords, use the following command:

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

Example: Set Password History Rules

You want to make users change their passwords at least every two weeks, and you want to prevent users from reusing any of their last 20 passwords.

However, you also want to prevent them from changing their password more than once a day to "flush" their password history.

To do this, use the following commands:

```
set password-policy = true;
set password-age = 14;
set password-history = 20
set password-min-age = 1;
```

Set the Number of Grace Logins

In a grace login system, passwords expire but users can use an expired password a limited number of times. This gives them the opportunity to change the password.

If the number of grace logins is exceeded, the account behaves as if it were expired.

The number of grace logins remaining is sent back to the binding client in the presence of the password policy request control.

To set the number of grace logins, use the following command:

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

If you use this command with an LDAP client that is aware of the Behera password policy request control, the client is informed of the number of logins remaining. Otherwise, the command works, but the client is not notified of the number of logins remaining.

Set Some Accounts to Never Expire

Accounts can be made to never expire. This is useful for accounts shared by many users, and for administrative or mission-critical accounts.

To set a user account to never expire

1. Set the *password-allow-ignore-expired* option to true by using the following command:

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

2. Add the attribute *dxPwdIgnoreExpired* to the user's entry.
3. Set the value of this attribute to *true*.

Note: To check which user passwords are set to never expire, search for all user accounts with the attribute *dxPwdIgnoreExpired = true*.

More information:

Set Some Accounts to Never be Suspended (see page 335)

Notify Clients of Expiring and Expired Passwords

CA Directory uses two password commands to mimic the way that Netscape directories work with LDAP password response controls.

To include LDAP response controls about password expiry to bind and compare responses, use the following command:

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

To set the number of days for which warnings about the password expiring are added to bind and compare responses, use the following command:

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

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.

During normal operation these commands cause bind and compare responses from a DSA to append a LDAP control containing the number of seconds before an account expires.

During the warning period, the password expiry notification control is appended to bind and compare responses.

More information:

Password Commands Requiring an LDAP Client (see page 321)

How to Set Account Suspension Rules

You can set a user account to be suspended if someone has made too many unsuccessful attempts to log in, or that user has not logged in recently.

You can configure both of these limits.

Suspend Unused Accounts

You can suspend accounts that have not been used recently. You can use the *set password-last-use* command to set the amount of time before an unused account is suspended.

If you use this command, each user must log in successfully at least as frequently as you set. If this does not occur, the account is suspended.

To set the number of days a password remains valid if it is not used, use the following command:

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

Example: Suspend Unused Accounts After 60 Days

In this example, you want to users to log in at least once every 60 days, and to change their password at least every 90 days.

To set this up, use these commands:

```
set password-policy = true;
set password-age = 90;
set password-last-use = 60;
```

Example: Suspend Unused Accounts, Do Not Force Password Changes

In this example, you want to allow people to keep using the same password, as long as they log in at least every 10 days.

To make passwords valid indefinitely except when they are not used for 10 days, use these commands:

```
set password-policy = true;
set password-last-use = 10;
```

You do not need to use the *set password-age* option because you are using its default value of 0 (off).

Suspend Accounts After Failed Login Attempts

To set accounts to be suspended after a certain number of attempts to log in, use the following command:

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

If you do not set this command, the default value of three retries is used.

Example: Allow Users Two Login Attempts

In this example, you want to allow users only two unsuccessful login attempts before suspending their account.

To set this up, use these commands:

```
set password-policy = true;
set password-retries = 2;
```

The following happens when a user tries to log in with an incorrect password:

1. Craig Link tries to log in using the wrong password.
2. He tries again with another incorrect password.

   His login is unsuccessful, and his account is suspended.

Set the Time after Which Users Can Attempt to Log In Again

If you use the *set password-retries* command to define how many times a user can attempt to log in before their account is suspended, you can also set a time after which the user can try again.

This means that after the period you set, the suspended account becomes active again.

If you do not use this option, an administrator must reset the user's password to unlock the account.

To let users try to log in again after a certain amount of time has passed, use the following command:

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

Example: Allow Users 5 Login Attempts, and a Delay of 30 Minutes before Trying Again

In this example, you want to allow users five unsuccessful login attempts before suspending their account. You then want to let the user try again after half an hour.

To set this up, use these commands:

```
set password-policy = true;
set password-retries = 5;
set password-max-suspension = 1800;
```

The following happens when a user tries to log in with an incorrect password:

1. Craig Link tries to log in using the wrong password.
2. He tries again four more times with incorrect passwords.

   His login is unsuccessful, and his account is suspended.
3. After 30 minutes, Craig Link's account is active again, and he can try to log in again.

Set Some Accounts to Never be Suspended

Accounts can be made to never be suspended, regardless of the number of incorrect login attempts.

You can use this to protect critical accounts that need to be immune to password lockout attacks.

To set a user's password to never be suspended

1. Use the following command to set the *password-allow-ignore-suspended* option to *true*:

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

2. Add the attribute *dxPwdIgnoreSuspended* to the user's entry.
3. Set the value of this attribute to *true*.

Note: To check which user passwords are set to never expire, search for all user accounts with the attribute *dxPwdIgnoreSuspended = true*.

More information:

Set Some Accounts to Never Expire (see page 330)

Example: Simple Password Policy

In this example, you want to set up a password policy with the following rules:

- Passwords can be reused. You do not want to keep a history of old passwords.
- A password must be at least eight characters long with at least one numeral.
- If a user's password is ever reset, you want the user to change the password immediately after the first login.
- If a password is not used for sixty days, you want the account to be suspended.

To set up this policy, set the following password rules:

```
set password-policy = true;
set password-min-length = 8;
set password-numeric = 1;
set password-force-change = true;
set password-last-use = 60;
```

Example: Complex Password Policy

In this example, the directory architect has created the following plain-English password policy:

- Passwords must contain at least seven characters, at least three of which must be alphabetical, and at least three of which must be numeric.
- No password may contain the user name.
- Passwords are valid for up to 14 days.
- Passwords must be at least one day old before they are changed, to prevent users from changing their password many times to fill up the password history.
- After a password has expired, users may still log in twice, to give them a chance to change the password.
- After three incorrect attempts at logging in, the account is suspended for 30 minutes, after which the user may try to log in again.

To implement this plan, use the following commands:

```
set password-policy = true;
set password-min-length = 7;
set password-alpha = 3;
set password-numeric = 3;
set password-username-substring = true;
set password-age = 14;
set password-min-age = 1;
set password-grace-logins = 2;
set password-retries = 3;
set password-max-suspension = 1800;
```

Enable Password Policies

Even if you have other password options set, they take effect only if you set the *password-policy* option to *true*.

To enable or disable password policies, set the following option:

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

Test Your Password Policy

To check whether a DSA supports the password policy control, you need to query the root DSE.

Example: Testing a Simple Password Policy

You can use this simple example to show that the CA Directory password policy is functioning correctly.

Client applications using this feature need to be able to parse the password policy response control. In test 1 and 2, the password policy response control is empty (no information to report).

1. In the Democorp directory, create the following password settings:

```
set password-policy = true;
set password-retries = 1;
```

2. Edit the Craig Link entry to include a password.
3. Use the LDUA to ensure that the DSA supports the password policy control:

```
ldua> bind-req
----> remote-addr = {                # server address
---->     psap = "PP"
---->     ssap = "SS"
---->     tsap = "TT"
---->     nsap = ip "hostname" port 19389
----> }
ldua> unbind-req;
```

```
ldua> search-req
----> base-object = <>
----> attrs = supportedControl;
ldua>
<- LDAP SEARCH-CONFIRM
invoke-id = 2   credit = 24
    Entry:    <>
    Contents:
    (supportedControl "1.3.6.1.4.1.42.2.27.8.5.1")
ldua>
```

Note: The *supportedControl* attribute is in the sunone.dxc schema.

- Use the following to bind with the password policy control:

```
ldua> bind-req
----> user =  <c au>
---->         <o Democorp>
---->         <ou Corporate>
---->         <ou Administration>
---->         <cn "Craig link">
----> password = "wrong"
----> remote-addr = {                # server address
----> psap = "PP"
----> ssap = "SS"
----> tsap = "TT"
----> nsap = ip "hostname" port 19389
---> }
---> controls = { password-policy };
ldua>
```

Test 1: Test with an Incorrect Password

1. Try to log in with an incorrect password.
2. The bind is refused.
3. The following response appears:

```
<- LDAP BIND-REFUSE
            invoke-id = 0   credit = 24
    Bind Error:    Security Error:  Invalid credentials
    Controls:
            password-policy response
```

Test 2 Test with an Incorrect Password Again

1. Try to log in again with an incorrect password.
2. The bind is refused again, and the account is suspended.
3. The following response appears:

```
<- LDAP BIND-REFUSE
            invoke-id = 0   credit = 24
    Bind Error:    Security Error:  Invalid credentials
    Controls:
            password-policy response
```

Test 3: Test with the Correct Password, but Account Suspended

1. Try to log in using the correct password.
2. The bind is refused because the account is suspended.
3. The following response appears:

```
<- LDAP BIND-REFUSE
            invoke-id = 0   credit = 24
    Bind Error:    Security Error:  Invalid credentials
    Controls:
            password-policy response
            Error: account-locked
```

User Account Administration

You can use password settings to lock and suspend user accounts.

Account locking

Use account locking when you need to manually lock and unlock an account.

Account suspension

A user account is suspended when the user tries to log in with the wrong password multiple times or does not change a password before it expires.

If an indefinite password suspension is required, then do not set *password-max-suspension* or set it to zero (re-init friendly).

Password management rules help you manage user accounts. The rules include the following:

- Forcing a user to change the password after the administrator has reset it
- Allowing administrators to lock a particular user's account
- Locking user accounts after a certain number of unsuccessful login attempts
- Setting the time after which users can try to log in again after locking due to unsuccessful login attempts
- Raising an SNMP trap every time an account is locked
- Nominating a user account to act as a single proxy user to handle all account administration

Reactivate a Suspended or Expired Account

User accounts can expire because the password is too old.

User accounts can be suspended for two reasons:

- The user has not logged in recently.
- The user has tried to log in with an incorrect password too many times.

Regardless of the reason that the account is inactive, an administrator can reactivate it by changing the password.

This works because most password rules are only applied when operations are being performed by the user who owns the password.

This means that whenever an administrator updates a user password, the password policy attributes for that entry are reset. These attributes include the following:

- The time of the last use of the password
- The time the password was created

To reactivate a suspended or expired account

1. Log in as an administrator.
2. Change the password of the suspended account.
3. Tell the user the new password.

   The user can now log in using the new password you created.

More information:

Enforce Password Rules When Reactivating an Account (see page 341)

Force Users to Change Passwords after Reactivation

You can force new users and users whose passwords have been reset to change their passwords.

To do this, use the following command:

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

When a user logs in with a newly reset password, the only operation the user can perform is to change passwords. After the user has changed his or her password, normal operations are restored.

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.

Do not use this feature if you have an application that performs a single bind to the directory for authentication. For these applications users would never be required to change their passwords.

When *set password-force-change* is set to *true,* DAP binds are refused because they cannot carry LDAP control information.

More information:

Reactivate a Suspended or Expired Account (see page 340)
Password Commands Requiring an LDAP Client (see page 321)

Enforce Password Rules When Reactivating an Account

Usually, when an administrator resets another user's password, the password quality rules are not applied. This means that the administrator can create passwords that would have been rejected if the users had tried to create the passwords themselves.

However, you can choose to enforce the password quality rules for all new passwords, including passwords set when an administrator reactivates a user's account.

To do this, set the following command to *true*:

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

This command means that all of the rules discussed in How to Configure Password Quality Rules (see page 323) and How to Set Account Expiry Rules (see page 328) apply to all password changes.

Lock a User Account

You can lock users' accounts manually by locking their password. You can later unlock the account, and the user can continue to use that password.

Note: If a user uses an LDAP client that is aware of LDAP password policy controls (for example, LDUA or a PAM-LDAP client), then the account-locked password policy control is returned in a bind refuse of a locked account.

To lock a user's account

1. Enable password locking using the following command:

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

2. Lock a user's account by adding the attribute *dxPwdLocked* with the value *true* to the user's entry.

More information:

Password Commands Requiring an LDAP Client (see page 321)

Unlock a User Account

To unlock a user account, remove the attribute *dxPwdLocked* from the user's entry, or set the value to *false*.

More information:

Lock a User Account (see page 342)

Check Which Accounts Are Locked

To check which user accounts are locked, search for all user accounts with the attribute *dxPwdLocked* with the value *true*.

The only way to find out if entries have this attribute is to search for the *dxPwdLocked* attribute explicitly. To actually return the *dxPwdLocked* attribute the attribute must be explicitly included in the entry information selection component of a search request.

Track Password Suspensions

You can track how often passwords are suspended for the following reasons:

- The password is too old
- The retry limit has been exceeded

There are two ways to track password suspensions:

Look in the alert log for *PASSWORD-SUSPENDED*

If the alert log is open and a password policy is active, one of these messages is written to the alert log every time a password is suspended.

Look in the SNMP log or use an SNMP aggregator

If the SNMP log is open and a password policy is active, you can choose to raise an SNMP trap every time a password is suspended, using this command:

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

Administer Passwords

CA Directory provides the tools to let you change the attributes of passwords, for example, minimum password length, and to change the password encryption.

How Password Encryption Works

When users add a password to their account, the following happens:

1. A user binds to a DSA.
2. The user adds a password to his or her user account.
3. The DSA encrypts the password, and then stores the encrypted password in the *userPassword* attribute.

   The encrypted password includes the name of the encryption scheme.

By default, DSAs use SHA-1 to encrypt passwords, but you can change to a different scheme if you prefer.

Choose an Encryption Method for Passwords Stored in a DSA

User accounts are entries in the directory with the *userPassword* attribute.

By default, the passwords stored in the *userPassword* attribute are encrypted using SHA-1.

However, you can use a different encryption scheme to encrypt these passwords. To do this, you need to create a password rule that specifies the encryption scheme. Each password is encrypted with the new scheme when it is next updated.

To choose a encryption method for passwords stored in a DSA

1. Ensure that the DSA is running.
2. Add the following command to the *dsaname.dxc* file:

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

3. Stop and start the DSA.

   Passwords that have already been encrypted are not updated automatically. The next time that a password is updated, it is encrypted using the new scheme.

Convert Passwords Already in a DSA to a New Encryption Method

If you change the encryption method for passwords in a DSA that already has encrypted passwords, these are not automatically updated. Although new passwords will be encrypted using the new method, existing passwords will use the old method.

To convert the encryption method for passwords already in a DSA

1. Choose an encryption method for passwords stored in a DSA (see page 344).
2. Search the DSA for all passwords encrypted using the old encryption method, and write the results to an LDIF file.
3. Open the LDIF file in a text editor and make the following changes:
   a. Add the text *dn:* to the beginning of each DN.
   b. Add the following lines below each DN:

   ```
   changetype: modify
   replace: userPassword
   userPassword: new-password
   ```

   Replace *new-password* with the actual new password for each entry.
4. Save the changed LDIF file.
5. Load the LDIF file into the DSA using the DXmodify tool.
6. Use the DXsearch tool to confirm that no passwords are encrypted using the old method, and also that passwords are now encrypted using the new method.

Example: Convert to the SSHA-1 Encryption Scheme

This example shows how to convert the passwords in the Democorp DSA from SHA-1 to SSHA-1.

In this example, the details are as follows:

- Old encryption scheme: SHA-1
- New encryption scheme: SSHA-1
- Democorp host: host23
- Democorp port number: 19389
- Name and password of the user who is authorized to change passwords: AdminUser, adminpassword

Follow these steps:

1. Ensure that the Democorp DSA is running.
2. Search the DSA for all passwords encrypted using SHA-1, using the following command:

   ```
   dxsearch -b "(o=democorp, c=au)" -s sub -D "{cn=AdminUser}" -w adminpassword -h
   host23 -p 19389 (userPassword={SHA}*) dn > sha-1.ldif
   ```

   This returns the entries in which the password is currently encrypted using SHA-1, and writes the results to the file *sha-1.ldif*.
3. Add the following command to the *democorp.dxc* file:

   ```
   set password-storage = ssha-1;
   ```
4. Stop and start the Democorp DSA.

   Each password is encrypted using the new encryption scheme the next time it is updated.

   To update all of the passwords now, follow the remaining steps.
5. Open the file *sha-1.ldif* in a text editor.
6. For each DN in the file, follow these steps:

   a. Add the text *dn:* to the beginning of each DN.

   b. Add the following lines below each DN:

   ```
       changetype: modify
       replace: userPassword
       userPassword: new-password
   ```

   Replace *new-password* with the actual new password for each entry.

7. Save the changed LDIF file.
8. Ensure that the Democorp DSA is running.
9. Load the file into the DSA using the following command:

   ```
   dxmodify -r -c -h host23:19389 -D "{cn=AdminUser}" -w adminpassword -f sha-1.ldif
   ```

10. Confirm that the passwords have changed by performing another search for SHA-1 hashed passwords, using the following command:

    ```
    dxsearch -b {Base DN for search} -s sub -D "{cn=AdminUser}" -w adminpassword -h
    host23:19389 (userPassword={SHA}*) dn"
    ```

    The search result should show that no passwords are now encrypted with SHA-1.

11. Confirm that the passwords are encrypted with SSHA-1 by repeating the search for SSHA-1 passwords:

    ```
    dxsearch -b {Base DN for search} -s sub -D "{cn=AdminUser}" -w adminpassword -h
    host23:19389 (userPassword={SSHA}*) dn"
    ```

Hide Passwords in Knowledge Files

The knowledge file for a DSA can contain the passwords for other DSAs and for LDAP servers, which allows the DSA to authenticate those other DSAs and LDAP servers. However, because the knowledge file is in plain text, this is a security risk.

The DXpassword tool can produce obfuscated passwords, by specifying the encryption method CADIR. This helps shield passwords in configuration files from users with access to the computer running the DSA.

To hide passwords in knowledge files

1. Identify the password of the remote DSA or LDAP server.
2. Use the DXpassword tool to encrypt the password using the CADIR option:

   ```
   dxpassword -P CADIR password
   ```

   The output contains the hashed password.
3. Copy the entire output into the knowledge file, replacing the clear-text password.
4. Save the file and stop and start the DSA.

Example: Hide an LDAP server's password

In this example, the password for the LDAP server is HelloThere. The DSA knowledge file currently contains the following line:

```
ldap-dsa-password = "HelloThere"
```

1. Use the DXpassword tool to encrypt the password using the CADIR option:

   ```
   dxpassword -P CADIR HelloThere
   ```
2. Note the output from the DXpassword tool:

   ```
   {CADIR}4YuDX1xmndSL7A==
   ```
3. Include the entire output in the knowledge file:

   ```
   ldap-dsa-password = "{CADIR}4YuDX1xmndSL7A=="
   ```
4. Save the knowledge file and stop and start the DSA.

Use a Password Proxy User

Some applications implement password policy by binding as a single user. All password comparisons and modifications are then performed by that user on behalf of all users. This can be used because SSL is expensive to establish, which means that allowing each user to create an SSL connection may be impractical.

This means that if someone binds as a different user for account administration reasons, password checks and changes to password policy are ignored.

To apply password policies to a proxy user

1. Create an account in the directory for the password proxy user.
2. Use the following command to identify this account as the password proxy user:

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

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

Operational Attributes for User Accounts

CA Directory uses operational attributes with some password settings, as shown in the following table.

You can use these operational attributes to diagnose problems with user accounts.

Password Command	**Operational Attribute**
set password-age command	*dxPwdLastChange*
set password-history command	*dxPwdHistory*
set password-min-age command	*dxPwdLastChange*
set password-grace-logins command	*dxPwdGraceLogins* *dxPwdGraceUseTime*
set password-allow-ignore-expired command	*dxPwdIgnoreExpired*
set password-allow-ignore-suspended command	*dxPwdIgnoreSuspended*
set password-force-change command	*dxPwdMustChange*
set password-last-use command	*dxPwdLoginTime*
set password-retries command	*dxPwdFailedAttempts*
set password-max-suspension command	*dxPwdFailedTime*
set password-allow-locking command	*dxPwdLocked*

Chapter 19: Connect to LDAP Clients

This section contains the following topics:

LDAP Clients

CA Directory DSAs can be accessed from LDAP clients, such as web browsers, address books, and Lightweight Directory User Agents (LDUAs).

Defining the LDAP Port

You define the LDAP address using the *set dsa* command, which contains a port number that listens for LDAP requests, as follows:

```
set dsa “Eagle” = {
    ...
    address = tcp “eagle” port 389
    ...
 };
```

This address definition is mandatory; it automatically enables LDAP as the DSA listens for different protocols on the same port.

For a list of all ports in a default installation of CA Directory, see Port Numbers Used by CA Directory.

Configuring LDAP Names

You configure LDAP names along with the usual schema attribute and object class definitions, as follows:

```
set attribute 2.5.4.3 = {
    name        = commonName
    ldap-names          = cn
    syntax              = caseIgnoreString
};
```

LDAP names are integrated into the DSA schema.

More information:

Set Up Schemas (see page 77)

Handling LDAP Strings

LDAP uses only string labels for information, so CA Directory resolves them to their true object and attribute identifiers by looking up their schema information. For each object and attribute label specified in the LDAP service requests, DXserver looks first for matching ldap-names, and then for a name.

LDAP is a string-handling protocol only; therefore, the following restrictions are implicit:

- Developers of directory clients must ensure the DSA receives an appropriately formatted LDAP message because the LDAP syntax is highly dependent on appropriate punctuation, white space, and so forth.
- Developers of LDAP directory clients must define locally customized attributes and objects through the DSA schema definition process.
- Developers should remember that LDAP names are not necessarily unique. For example, two organizations can define the string *mail* but have different syntaxes and uses for it.

Transparent Routing

When using a router DSA to route client requests to external LDAP directories using DXlink, it may not be feasible or convenient to include all third-party schema.

Schema Publishing

CA Directory supports schema publishing through LDAP Version 3. This enables LDAP directory clients to read the directory schema dynamically from the DXserver using LDAP Version 3. For further information, see section 4.2 Subschema Subentries in RFC 4512.

The following information is published through the *cn=schema* subschema subentry using LDAP Version 3:

- Attributes
- Object Classes
- Attribute Types
- Syntaxes
- Name Forms
- Structure Rules

This schema information can be retrieved by performing a base-object search of the *cn=schema* subentry with a search filter of *objectClass* present.

LDAP Controls

LDAP controls are a mechanism for extending or changing the way a server handles a particular operation. Controls may be attached to LDAP operations and generally change the semantics for that specific operation. Controls that are not recognized are ignored if they are not marked *critical*. If a control is marked *critical* and is not recognized, the error *unsupported critical extension* is returned and the operation is not performed.

Because CA Directory supports distribution, it is possible that a control is not honored even though it is recognized. For example, if server-side sorting is requested but the request has to be multi-chained to complete the operation, the server-side sorting control is not honored. If the control is marked *critical*, an error is returned to the client; otherwise, the control is ignored.

Controls recognized by CA Directory are published through the root DSE according to RFC 4512.

Server-Side Sorting

The server-side sorting LDAP control allows a client to request that a search response be sorted.

See RFC 2891.

Simple Paged Results

The simple paged results LDAP control allows a search result to be returned a page at a time. It is only available when DXcache has been enabled.

See RFC 2696.

Password Policy Control

The Password Policy control lets applications that manage accounts receive extra information on operation responses. The information returned through this control includes account status (password expiring or expired, account locked) and reasons for password change failure (password too short, and so on).

This control is specified in an Internet Draft on the IETF home page. The specification of its operation can change over time. Also, the name of the draft document changes as revisions are made. At the time of writing, the document name is *draft-behera-ldap-password-policy-08.txt*.

Proxied Authorization Control

The proxied authorization control enables a client to perform operations as the user specified in the control value. This is similar to the ‘su’ (substitute user) function available on UNIX operating systems. To restrict the ability of a user to impersonate anyone, the *trust SASL proxy* feature must be set. This feature is useful for auditing changes to the DIT.

If the link flag *dsa-ldap-proxy* is used with DXlink, the proxied authorization control is added to operations chained to third-party LDAP servers containing the entry of the request's originator. The root DSE of the third-party LDAP server should be queried to check that it supports this control.

This control is specified in an Internet Draft on the IETF home page. The specification of its operation can change over time. Also, the name of the draft document changes as revisions are made. At the time of writing, the document name is *draft-weltmann-ldapv3-proxy-13.txt*.

Netscape Password Policy Controls

The Netscape password policy controls are used to notify an application that the connecting user's password is expiring (number of seconds to expiry) or has expired. These are response-only controls and are enabled in CA Directory with the following command:

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

These controls are described in an Internet Draft that has expired. They are not subject to IETF standardization.

Enable Persistent Search

This control allows a client to request notification when entries in a specified scope change.

This control can be used only if the scope it specifies is mastered by the DSA it is sent to. This is because the DSA does not coordinate with other directories (including CA Directory DSAs) holding the same or other parts of the DIT.

To enable persistent searching, use the following command:

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

The control is specified in an Internet Draft that has expired. It is not subject to IETF standardization.

For more information, see draft-ietf-ldapext-psearch-03.txt on the IETF home page.

Limitations on Persistent Searching

Persistent searching can be used only where the namespace is managed by a single DSA. Only the local namespace is accessed, which means that there is little point in using it with routers.

For example, persistent searching does not work in the following situations:

- If you have update and search DSAs, then the persistent search is not likely to work unless it is done in the router.
- If there are multiple routers, then persistent search in the router does not work. Instead, it must be done on the update DSA.
- If there is query streaming and persistent search is done in the update DSA, the search is never sent to the update DSA, so it does not work in this situation either.

Routing LDAP Controls

If CA Directory has to chain (forward) a request to another directory, it also forwards any associated controls, even if it does not recognize them. It does not forward these controls when multicasting a search request.

Controls on responses are also passed back to the client even when not recognized, if the result is not an error.

LDAP Root DSE

The top of the directory information tree (DIT) on each DSA is the *root* of the tree.

The root contains a special entry called the root DSA-specific entry (DSE). This is specified in the LDAP [RFC4512] and therefore can only be read using the LDAP protocol.

The information provided by reading the root DSE is useful for application developers in determining information, features, and schema specific to the LDAP directory.

The following examples to retrieve root DSE information use the *ldapsearch* utility available on most operating systems.

Display the Whole Root DSE

To display the entire root DSE, use the following command:

```
ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*"
```

Display the LDAP Version

To display the version of LDAP supported by the DSA, use the following command:

```
ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*"
supportedLdapVersion
```

For example, the output could look like the following:

```
supportedLdapVersion=3
```

Display the CA Directory Version

The root DSE of CA Directory DSAs includes the attribute *dxServerVersion*.

The presence of this attribute indicates to clients that this is a CA Directory DSA. The value of the attribute contains version information.

Use the following command to check the CA Directory version:

```
ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*" dxServerVersion
```

If no result is returned, the DSA is not a CA Directory DSA.

Display the Supported LDAP Controls

To list the LDAP controls supported by the DSA, use the following command:

```
ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*" supportedControl
```

The following table describes some LDAP controls supported by CA Directory DSAs:

Control	Description	Comments
supportedControl=1.2.840.113556.1.4.473	Server Side Sorting	
supportedControl=1.2.840.113556.1.4.319	Simple Paged Results	
supportedControl=1.3.6.1.4.1.42.2.27.8.5.1	Behera Password Policy	
supportedControl=2.16.840.1.113730.3.4.3	Persistent Search	Only supported after it is enabled.
supportedControl=2.16.840.1.113730.3.4.18	Proxy Authorization	

Display the Naming Context

The following command lists the prefix of the DSA:

```
ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*" namingContexts
```

Display the Supported Schema

The subschema subentry is a virtual entry containing the schema being currently used by DSA.

Display the schema supported by this DSA

1. Use the following command:

   ```
   ldapsearch -h hostname -p portnumber -s base -b "" objectclass="*"
   subschemaSubentry
   ```

   The information retrieved is the subentry that contains the schema information, for example:

   ```
   subschemaSubentry=cn=schema
   ```

2. Perform another search from this base to retrieve the schema, for example:

   ```
   ldapsearch -h hostname -p portnumber -s base -b "cn=schema" objectclass="*"
   ```

   This returns the following information:

   - objectClasses
   - attributeTypes
   - dITStructureRules
   - nameforms
   - ldapSyntaxes

LDAP Programming in Perl

This section illustrates how to access LDAP data using the Net::LDAP module, which is available from any Perl CPAN archive.

This section assumes that you are familiar with Perl; it does not cover the installation of Perl or the Net::LDAP module. This example is provided without any implied or explicit warranties. CA does not provide support for either the Perl language or the Net::LDAP Perl module.

Note: The line numbers reference the comments (they are not part of the program).

This program extracts entries with the job title *Secretary* from the Democorp directory, and prints their name and telephone number:

```
#!/usr/local/bin/perl -w

# This is a demonstration of accessing the Democorp directory using the
# Net::LDAP Perl modules
use strict;
use warnings;
use Net::LDAP;

my($ldap, $mesg, $entry);

$ldap = Net::LDAP->new('yourhost.com:19389') or die "Can't connect: $@";

# anonymous bind
$mesg = $ldap->bind;
if($mesg->is_error){
 die "Bind error: ".$mesg->error;
}

# search for job title = "Secretary"
$mesg = $ldap->search(base => "o=Democorp,c=AU",
                   filter => "(title=Secretary)"
                   );
if($mesg->is_error){
 die "Search error: ".$mesg->error;
}

foreach $entry ($mesg->entries) {

 # report entry's common name (cn) and telephone number
 print $entry->get_value('cn'), "\t",
    $entry->get_value('telephoneNumber'), "\n";

}

print "Finished\n";

$ldap->unbind;
```

Comments on LDAP Program

Line 13

Replace yourhost.com with the name of the machine on which the Democorp directory is installed and running.

Line 16

This is an anonymous bind. There is no DN or password supplied.

Line 17

Always check the results of the requests.

Line 22

Execute the query and check for any errors.

Line 29

This loop executes for each entry found by the search. No errors are possible at this point.

Line 32 and 33

Print out the entries found. This usually produces the following output:

```
Juliet LEVY   524 3637
Craig LINK    544 3697
Terry SALISBURY 488 0288
Winifred LEWIS  534 3657
Ian NATHAN      437 8908
Cheryl APPLEBY  498 0308
Finished
```

Line 39

Close the connection.

Chapter 20: Troubleshooting CA Directory

This section contains the following topics:

Cannot Create a DSA

Symptom:

I cannot create a DSA from the Namespace Map.

Solution:

If you cannot create a DSA from the Namespace Map, you may need to clear the Java Applet Cache.

To clear the Java Applet Cache:

1. Close the DXmanager browser window.
2. Select Start, Settings, Control Panel.
3. In the Control Panel, double-click on Java.
4. On the General tab, click the Settings button under Temporary Internet Files.
5. Click on the Delete Files button.
6. Ensure the Applications and Applets checkbox is selected.
7. Click OK.

The cache is now cleared and the DXmanager browser can be opened again.

Cannot Start a DSA

Symptom:

I cannot start a DSA.

Solution:

If a DSA fails to start, work through the following steps to diagnose the problem:

- Has your computer been in standby mode?

  If so, restart it to regain the TCP/IP stack.

- Does your computer have a network card and an IP address assigned?

  If not, install your network card and assign an IP address (or use DHCP).

- Check the error messages. For more information, see System Messages.

  Check the following server logs:

  - DXHOME/logs/dsa-name_alarm.log
  - DXHOME/logs/dsa-name_trace.log

- For assistance, contact Technical Support at http://ca.com/support

Collate Diagnostic Information for CA Support

When you log a support issue for CA Directory, you may be asked to supply information about your directory. CA Directory includes DXinfo, which is a tool that compiles this information for you.

If your issue involves distribution, replication, or remote operations, you should send CA Technologies information from all directory hosts. This helps our Support staff develop a complete picture of your directory backbone.

To collate information for CA Support

1. Prepare the directory, by following these steps on each directory host:

   a. Back up the CA Directory logs to a safe location, and then remove the original log files. The log files are in DXHOME/logs.

   b. Use Telnet to connect to the DSA. For example, to connect to the Democorp DSA on the local computer, use the following command:

   ```
   telnet localhost 19389
   ```

   c. Check the levels of trace that is being logged, using the following command on the DSA console:

   ```
   get trace;
   ```

   d. Record the trace levels. You need to restore these later.

   e. Set the trace level to capture all events. To do this, use the following command on the DSA console:

   ```
   set trace=full;
   ```

2. Reproduce the problem. Once is sufficient.

3. Collate the information needed by CA Support, by following these steps on each directory host:

   a. (UNIX only) Log in as the user *dsa*.

   b. Use the DXinfo tool to capture the required information, using the following command:

      ```
      dxinfo
      ```

      For the full syntax of this command, see DXinfo Tool—Collect System Information.

      The DXinfo tool creates a series of files in the current working directory.

   c. (Optional) Edit the files to remove any sensitive information that you do not want to send to CA Technologies.

   d. If DXDUMPCORE is defined, get the latest core dump file from DXHOME.

      The core dump file is named *core* (*core.processID* on Linux systems).

4. Send the files created by the DXinfo tool, and the core dump file if there is one, to Technical Support for analysis.

Collecting Data if a DSA Fails Internal Tests—DXDUMPCORE Environment Variable

The DSA program contains internal tests ("assertions") that it runs to ensure that the program is running correctly.

If one of these internal tests fails, it means that the DSA is in an unexpected state, and this should not happen. However, it does happen, though rarely.

Normally if an internal test fails, the DSA logs this and continues to run. This behavior ensures that the DSA does not stop, which is what you want when the DSA is used in a production environment. However the log entry might not contain enough information to fully analyze the problem that caused the test to fail.

To collect more information, and so enable a full analysis of the problem, define the environment variable DXDUMPCORE on the DSA host. If DXDUMPCORE is defined and an internal test fails, the DSA writes the contents of the program to a file (sometimes this is called a core dump) and then stops. The file that contains the core dump is in DXHOME and is called:

- core (All systems except Linux)
- core.*ProcessId* (Linux Systems)

Important! Only define DXDUMPCORE if you are prepared for the DSA to stop if it fails an internal test. Make sure DXDUMPCORE is not defined in production environments, because if it is defined, the DSA is vulnerable to Denial of Service attacks.

If DXDUMPCORE is defined and the DSA fails an internal test, you can send the core dump file (with the DXinfo output) to CA Support for a full analysis of the problem.

Appendix A: Set Up the Sample Directories

This section contains the following topics:

Sample Directories

CA Directory provides two sample directories. Use these sample directories to explore CA Directory before setting up your own directory.

- **Democorp** – The Democorp sample directory is an example of a corporate staff directory. It includes the details of about 1200 staff members, arranged in about 100 organizational units.
- **UNSPSC** – The UNSPSC sample directory includes all of the products and services in the Universal Standard Products and Services Classification (UNSPSC). This directory contains more than 10,000 entries.

Each directory comes as a LDIF file plus a script that converts the data in the file into a datastore.

You can then use DXmanager to set up DSAs that use those datastores.

You can also examine the scripts that set up the samples, which illustrate how to load data into your own directory.

How the Sample Directory Scripts Work

Each of the the setup scripts for UNSPSC and Democorp performs the following steps:

1. Converts the data from CSV format into LDIF, using the csv2ldif tool.
2. Loads the LDIF entries into the datastore using the DXloaddb tool.

Note: in order for the scrips to work, the DSAs need to be created first. You can do this using either the DXmanager of the dxnewdsa tool.

How to Set Up the Sample Directories

When you install the Directory package, the files for the sample directories are automatically installed.

To set up the Democorp and UNSPSC directories using DXmanager, you need to follow these steps:

1. Set up the Democorp namespace partition (see page 369).
2. Set up the UNSPSC namespace partition (see page 372).
3. Set up the root namespace partition (see page 374).
4. Add more hosts to the sample directory backbone (see page 376).
5. Create Democorp DSAs on the new hosts (see page 377)
6. Create UNSPSC DSAs on the new hosts (see page 378).
7. Create a router DSA on each host (see page 379).
8. Deploy the backbone configuration to the hosts (see page 381).
9. Source the UNSPSC Schema (see page 382)
10. Set up the sample datastores on the hosts (see page 383).
11. Update the sample directory configuration files (see page 384).

Set Up the Democorp Namespace Partition

DXmanager lets you create configuration for one or more directory hosts. These instructions describe how to set up configuration for a simple backbone that includes one region, one site, and three hosts.

Note: This description assumes that you use DXmanager the first time you set up a namespace otherwise the wizard will not appear.

To set up the Democorp namespace partition

1. Log in to DXmanager as a user with the Change or Superuser role.
2. Select the Single DSA Backbone link.
3. Use the wizard to enter the following default values for the backbone:

   Console password

   Specifies the password required for connection to this DSA on a DSA console. You must specify a password if you want to connect to this DSA from a remote computer. This password is optional for local connections.

   Password

   Specifies the password that DXmanager uses when contacting DXadmind on the host. This password is used for communication between DXadmind and DXmanager.

   Ensure that you enter same password that was specified when CA Directory was installed on each host.

4. Use the wizard to enter the following namespace partition details:

   Partition name

   Specifies the name of the namespace partition that you are creating. This name is used on the DXmanager maps.

   Enter: Democorp

   Prefix

   Specifies the context prefix of the DIT that the DSA publishes that it serves. If *Native prefix* is also set, then this becomes the logical prefix rather than the physical prefix and prefix mapping is done in the server.

   Enter: o=Democorp,c=au

Port

Specifies the port for this namespace partition. The DSAs created when you deploy this namespace partition to a host use this port as their listen address.

Enter: 19389

SNMP port

Specifies the SNMP port for this partition. This is the port the DSA will use to send SNMP traps. This is needed by the DXmanager to be able to monitor DSAs.

Enter: 19389 (automatically populated when you enter the Port).

5. Create the topology. This wizard lets you create a single-host topology only, which means that you can create only one region, one site, and one host. You can add more later. Enter the following details:

Region

Specifies the name of the region.

Enter: Australia

Site

Specifies the name of the site.

Enter: Melbourne

Host name

Specifies the name of the host.

Enter: *Your host name*

Data Store Location

Identifies the location of the data store. If a relative path is given it is relative to DXHOME.

Enter: data

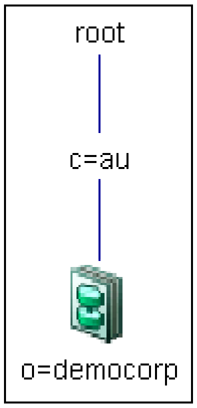
Note: the data directory is created by default when CA Directory is installed.

Network Address

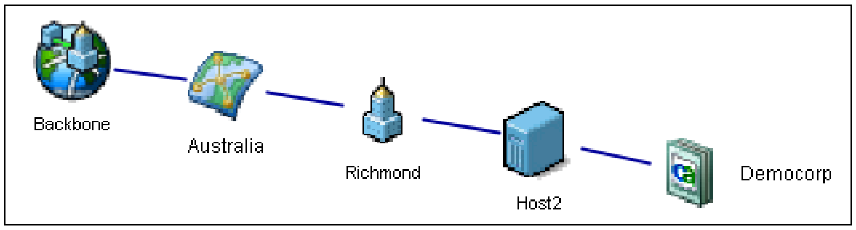
Identifies the host on the network, in one of the following formats: IP address or network address.

6. When you are satisfied with the settings, click Finish.

   The Maps tab appears, showing the Namespace map. It should appear similar to the following:



   You can also look at the new Topology map, which should appear similar to the following:



   These maps have been created from the details you just entered.

7. Save this version of the configuration, as follows:

   a. Click Configuration, Save.

   b. Enter a comment that describes the current state of the configuration, and then click OK.

   You can now set up the UNSPSC namespace partition (see page 372).

Set Up the UNSPSC Namespace Partition

These instructions describe how to add the UNSPSC namespace to the existing Democorp DIT.

To set up the UNSPSC namespace partition

1. Navigate to the Namespace map.
2. Right-click the c=au node and select Add new partition.
3. On the General tab, enter the following details:

   Name

   Specifies the name of the namespace partition that you are creating. This name is used in DXmanager displays.

   Enter: UNSPSC

   Prefix

   Specifies the context prefix of the DIT that the DSA publishes that it serves. If *Native prefix* is also set, then this becomes the logical prefix rather than the physical prefix and prefix mapping is done in the server.

   Enter: o=UNSPSC,c=au

4. On the Connections tab, enter the following details:

   Port

   Specifies the port for this namespace partition. The DSAs created when you deploy this namespace partition to a host uses this port as their listen address.

   Enter: 19489

5. On the Console tab, enter the following details:

   Console port

   Specifies the port on which the DSA accepts connections from the local computer only. If this is not specified, there is no console for the DSA.

   Enter: 19490 (automatically populated when you enter the Port).

   Remote console port

   Specifies the port on which the DSA accepts connections from remote computers.

   Enter: 19491 (automatically populated when you enter the Port).

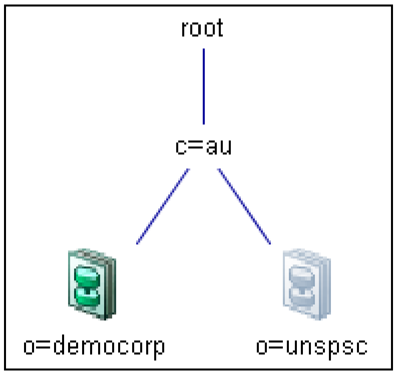
6. On the Monitoring tab, enter the following details:

   SNMP Port

   > Specifies the SNMP port for this partition. This is the port the DSA will use to send SNMP traps. This is needed by the DXmanager to be able to monitor DSAs.
   >
   > **Enter:** 19489

7. When you are satisfied with the settings, click OK.

   The Maps tab appears, showing the Namespace map. It should appear similar to the following:



8. Save this version of the configuration, as follows:

   a. Click Configuration, Save.

   b. Enter a comment that describes the current state of the configuration, and then click OK.

   You can now set up the Root namespace partition (see page 374).

Set Up the Router Namespace Partition

You have now set up namespace partitions for the Democorp and UNSPSC DSAs. We recommend that you now set up a router partition.

To set up the Routernamespace partition

1. Navigate to the Namespace map.
2. Right-click the c=au node (middle of the tree) and select Add new partition.
3. Enter the following details:

   Partition name

   Specifies the name of the namespace partition that you are creating. This name is used in DXmanager displays.

   Enter: router

   Prefix

   Specifies the context prefix of the DIT that the DSA publishes that it serves.

   Enter: c=au (automatically populated)

   Port

   Specify the port for this namespace partition. The DSAs created when you deploy this namespace partition to a host uses this port as their listen address.

   Enter: 19289

   Console port

   Specifies the port on which the DSA accepts connections from the local computer only. If this is not specified, there is no console for the DSA.

   Enter: 19290 (automatically populated when you enter the Port).

   Remote console port

   Specifies the port on which the DSA accepts connections from remote computers.

   Enter: 19291 (automatically populated when you enter the Port).

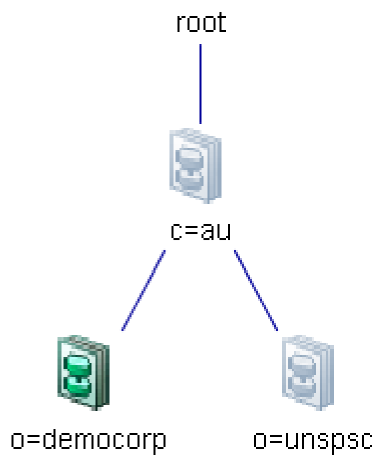
   SNMP port

   Specifies the SNMP port for this partition. This is the port the DSA will use to send SNMP traps. This is needed by the DXmanager to be able to monitor DSAs.

   Enter: 19289

4. When you are satisfied with the settings, click Finish.

   The Maps tab appears, showing the Namespace map. It should appear similar to the following:



5. Save this version of the configuration, as follows:

   a. Click Configuration, Save.

   b. Enter a comment that describes the current state of the configuration, and then click OK.

   You can now add more hosts to the directory backbone (see page 376).

Add More Hosts to the Sample Directory Backbone

You now need to create configuration for two more hosts.

To add more hosts to the sample directory backbone

1. Ensure that DXmanager is still in Edit mode.
2. Navigate to the Topology map.
3. Right-click the Site icon, and select Add a new host to the site.
4. Enter the following details:

   Host name

   Specifies the name of the new host.

   Network Address

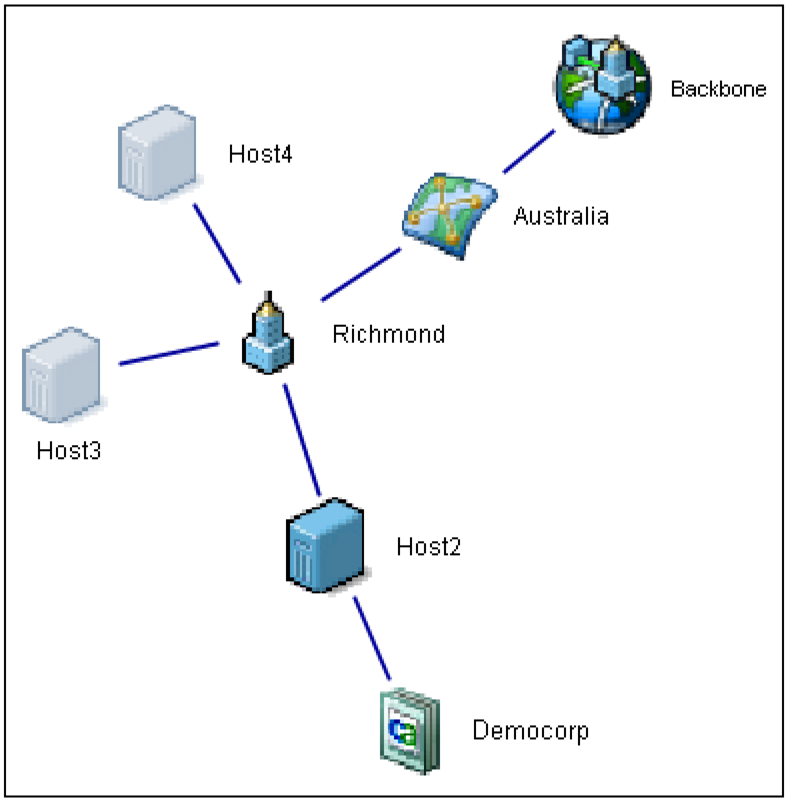
   Identifies the host on the network, in one of the following formats: IP address or network address.

5. Click Finish.

   You have now created the configuration for a new host.

6. Repeat Steps 1-3 to create the third host.

   The Topology map should now appear similar to the following:



7. Save this version of the configuration, as follows:

    a. Click Configuration, Save.

    b. Enter a comment that describes the current state of the configuration, and then click OK.

    You can now create Democorp DSAs on the new hosts (see page 377).

Create Democorp DSAs on the New Hosts

When you used the Single-DSA wizard to create the Democorp backbone, you configured a Democorp DSA on one host. You then created two more hosts, without creating a DSA on them. The following instructions describe how to create Democorp DSAs on the two new hosts.

To create Democorp DSAs on the new hosts

1. Navigate to the Topology map.

    Right-click a host, select Instantiate a new partition to this host, and then select As a Data DSA.

2. The Instantiate Namespace Partition as a Data DSA page opens.

3. Select the Democorp option in the Namespace Partition list.

    The details should be correct.

4. Click Next, and then review the list of network addresses for this host. If the list is correct, click Finish.

5. Repeat these steps for the third host.

6. Save this version of the configuration, as follows:

    a. Click Configuration, Save.

    b. Enter a comment that describes the current state of the configuration, and then click OK.

    You can now create UNSPSC DSAs on the new hosts (see page 378).

Create UNSPSC DSAs on the New Hosts

You have now created Democorp DSAs on all hosts. You should now create a UNSPSC DSA on each host.

To create UNSPSC DSAs on the new hosts

1. Navigate to the Topology map.
2. Right-click a host, select the select Instantiate a new partition to this host, and then select As a Data DSA.

   The New Relationship dialog opens.
3. Select the UNSPSC option in the Namespace list.
4. Click Next, and then review the list of network addresses for this host. If the list is correct, click Finish.
5. Repeat these steps for the other two hosts.
6. Save this version of the configuration, as follows:
   a. Click Configuration, Save.
   b. Enter a comment that describes the current state of the configuration, and then click OK.

   You have now created configuration for all three DSAs in this backbone.

Create a Router DSA on Each Host

You have created two data DSAs for each host. You should now create a router DSA on each host.

To create a router DSA on each host

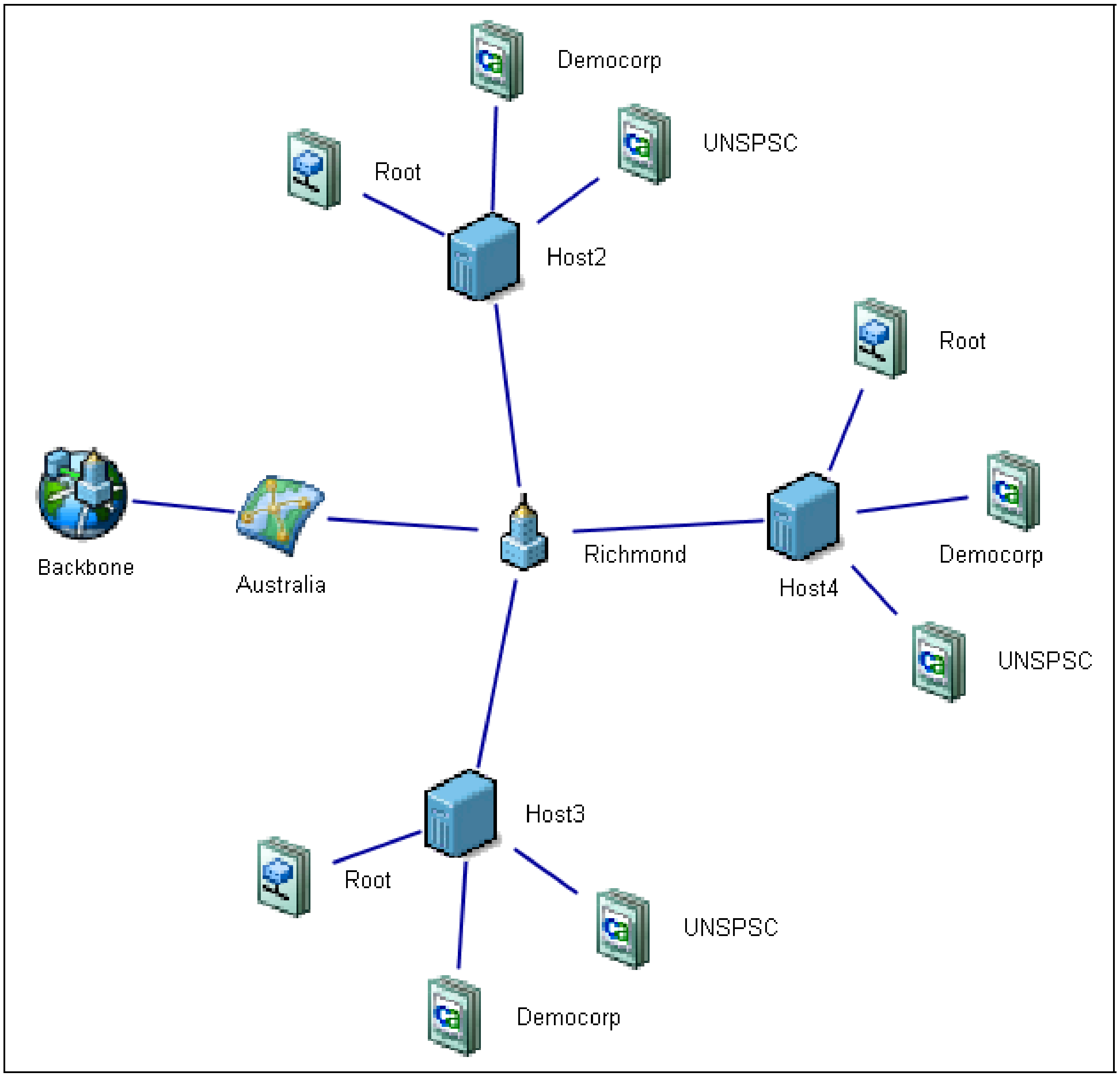
1. Navigate to the Topology map.
2. Right-click a host, select Instantiate a new partition to this host, and then select As a Router DSA.
3. The Instantiate Namespace Partition as a Data DSA page opens.
4. Select the Router option in the Namespace Partition list.

   The details should be correct.
5. Click Next, and then review the list of network addresses for this host. If the list is correct, click Finish.
6. Repeat these steps for the other two hosts.
7. Save this version of the configuration, as follows:
   a. Click Configuration, Save.
   b. Enter a comment that describes the current state of the configuration, and then click OK.

   You have now created configuration for three router DSAs in this backbone.

   You can now deploy the backbone configuration to the hosts (see page 381).

The Topology map should now appear similar to the following diagram.


Democorp
Root
UNSPSC
Host2
Root
Backbone
Australia
Richmond
Democorp
Host4
UNSPSC
Host3
Root
UNSPSC
Democorp

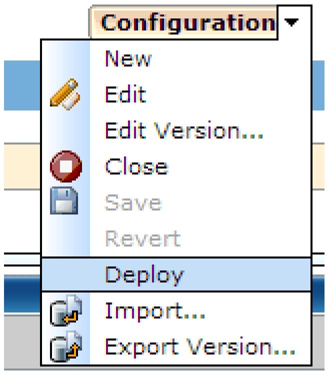
Deploy the Backbone Configuration to the Hosts

You have used DXmanager to create configuration for a directory that uses three hosts. The configuration is currently stored in a single XML file on the directory management server (the computer on which DXmanager is running).

You now need to deploy the configuration to the hosts. This process sends a copy of the configuration file to each host and creates any required DSAs.

To deploy the backbone configuration to the hosts

1. Find the Configuration button at the top right of the DXmanager window.
2. Click Configuration, Deploy, as follows:

   DXmanager sends the configuration file to each host, and each host creates the required DSAs.
3. Right-click the backbone icon, and then select Start all DSAs.

   All of the DSAs in the backbone start. Their icons change to show the green Started symbol.

   You can now update the configuration files (see page 384).

Source the UNSPSC Schema

DXmanager creates an initialization file for each new DSA, which sources the default configuration files, including some schemas.

However, the data in the UNSPSC DSA requires the UNSPSC schema, which is supplied with CA Directory.

To source the UNSPSC schema

1. Find the initialization file for the UNSPSC DSA in the following folder:

   ```
   DXHOME/config/servers
   ```

   The files each have the same name as the DSA, as in the filename *UNSPSC-Host1.dxi*.

2. Open the initialization file in a text editor and find the statement that sources a schema group file. For example:

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

3. Open the sourced group file in a text editor.

4. Add the following line to the file:

   ```
   source "unspsc.dxc";
   ```

5. Save and close the schema.

   When the UNSPSC DSAs are created, they all source the UNSPSC schema.

   You can now create a router DSA on each host (see page 379).

Set Up the Sample Datastores on Each Host

CA Directory comes with scripts that set up the sample Democorp and UNSPSC datastores. You can do this before or after creating the configuration in DXmanager.

To set up the sample datastores on each host:

1. Ensure that the Directory package is installed on the host.
2. Open a command prompt and navigate to the following location:

   ```
   DXHOME/samples/democorp
   ```

3. Run the setup script using one of the following:

   - For UNIX use:

     ```
     setup.sh democorp
     ```

   - For Windows use:

     ```
     setup.bat democorp
     ```

   This script creates the *democorp* datastore and populates it with sample data.

4. Navigate to the following location:

   ```
   DXHOME/samples/unspsc
   ```

5. Run the setup script using one of the following:

   - For UNIX use:

     ```
     setup.sh unspsc
     ```

   - For Windows use:

     ```
     setup.bat unspsc
     ```

   This script creates the *unspsc* datastore and populates it with sample data.

You can now set up the Democorp namespace partition (see page 369).

More information:

How the Sample Directory Scripts Work (see page 368)

Update the Sample Directory Configuration Files

Each DSA that you have just created has an initialization file. This file sources other configuration files, which contain settings for access controls, schema, password policies, and other settings not included in DXmanager.

You can edit each DSA's initialization file to source different configuration files. You can also edit the configuration file to include any custom settings you may need.

To update the sample directory configuration files

1. On one of the hosts, navigate to the folder that contains the initialization files:

   ```
   DXHOME/config/servers
   ```

   Each DSA's initialization file is contained here.

2. Open the initialization file in a text editor, change the source statements to point to different configuration files, and then save the file.

3. On the same host, navigate to the folders that contains the configuration files that the initialization file sources:

   ```
   DXHOME/config/access
   DXHOME/config/limits
   DXHOME/config/logging
   DXHOME/config/schema
   DXHOME/config/settings
   ```

4. Open a configuration file in a text editor, change the configuration commands, and then save the file.

5. Repeat Steps 1-3 for each DSA on each host.

6. Restart or reinitialize each DSA whose configuration you changed.

   The DSAs now use the new configuration.

Appendix B: Rarely-Used Features

This appendix describes how to use some special features of CA Directory. These specialized features are useful only for particular scenarios. There is no need for most customers to use them. Ensure that you actually need one of these features before you implement it.

This section contains the following topics:

Set Up Class-of-Service Templates

We recommend that you use views instead of class-of-service templates. Views provide the same functionality but allow for data to be stored in the directory rather than files. This protects your data and allows for backups, replication, and maintenance.

However, views do have some limitations in how information is retrieved. You could use class-of-service templates instead of views if you cannot change the applications to support the data access limitations.

When directory information needs to be shared across multiple entries, the shared information can be stored in an class-of-service template. A *class-of-service template* stores information that can then be included in many entries. Class-of-service templates can reduce the size of a directory, help keep data consistent, and reduce the time required for bulk updates.

When a search returns an entry that contains an attribute from the class-of-service template, the attributes and values from the associated template are included in the entry.

This means that the information in class-of-service templates is stored only once, which is good for three reasons:

- Directory size is reduced
- Simple bulk updates are faster
- Information is consistent

The shared information is usually a level of service, such as a standard or premium subscription to an Internet Service Provider.

How Class-of-Service Templates Work

The virtual attributes and values are stored in templates, which are kept in a DSA configuration file.

When a search returns an entry that uses a class-of-service template, the attributes and values from the associated template are presented.

A template can have multiple attributes and values. All attributes in a template are added to the entry.

If an entry already has a value for an attribute in a template, the attribute in the template can either overwrite the value, or the value can be left untouched. This is controlled by the disposition of the template attribute. The two possible dispositions are as follows:

default

The value from the template replaces the existing value.

override

If the entry already contains this attribute, the existing value persists. Use this when a customer requires special treatment.

The attributes that are stored in class-of-service templates are not searchable.

Working with Class-of-Service Templates

When the shared attribute values change, they can be updated in the configuration files. Therefore, ensure that configuration files are stored in a source code control system to permit rollbacks.

Distribution of class-of-service templates is achievable, but requires the DSA configuration to be manually copied to all nodes.

Configure the DSA to Allow Class-of-Service Templates

Add the following line to the local DSA configuration to clear any currently cached template information.

```
clear class-of-service;
```

This lets you change attributes while the DSA is still running, using the command *dxserver init*.

Ensure that the template configuration file is sourced after the schema is sourced. The template uses attributes that are defined in the schema configuration.

Add a Class-of-Service Template to an Entry

To use class-of-service templates with an entry, add the *cos-attr* attribute to the entry.

Create Class-of-Service Templates

Templates can be stored in any directory. However, if there are many templates, you should store them in separate files in the DXHOME/config/settings directory.

The class-of-service templates use the following syntax:

```
<cosTemplate>   ::= set class-of-service <cosLabel> = { <operCos> };
<cosLabel>      ::= <string>
<operCos>       ::= object-class = <attrOid> cos-attr = <attrOid> cos-value = <value>
attribute-values = { <cosTemplateList> }
<cosTemplateList>       ::= <cosTemplate> | <cosTemplate>, <cosTemplateList>
<cosTemplate>   ::= ( type = <attrOid> value = <cosValueList> disposition =
<cosDisposition> )
<cosValueList>  ::= value | value, <cosValueList>
<cosDisposition>::= default | override
```

Note: The *cos-attr* used in a class-of-service template must be a single-valued attribute.

View Class-of-Service Templates

To view templates in use in a DSA, use the following command from the DSA console:

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

This produces a listing of each class-of-service template in use, with the template label at the top.

Example: Output from the *get class-of-service* Command

The template for the standard class of service at Excellent ISP appears, as follows:

```
************** 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
```

Example: The Excellent ISP Company

The company Excellent ISP is an Internet service provider. The customers of Excellent ISP can subscribe for one of two classes of service described in the following table:

	Standard	**Premium**
Mail Storage	20 MB	30 MB
Web Space	20 MB	30 MB
Hours per Month	15 hr	Unlimited
Cost per Month	$19.95	$29.95
Cost of Extra Hours	$1.00/hr	$0.00/hr

The Excellent ISP customer directory includes the subscription information in each customer entry.

Example Entries without Class-of-Service Templates

Before you start using class-of-service templates, this information is stored in each entry.

If there are one million users, then these attributes appear one million times. If Excellent ISP increases its fees, then a large global update is required.

Here is an example entry for an Excellent ISP customer who has the standard class of service:

```
dn: cn=John Smith, o=Excellent ISP, c=US
oc: inetOrgPerson
oc: excellentISPuser
cn: John Smith
sn: Smith
excellentISPmailQuotaMB: 20
excellentISPwebSpaceMB: 20
excellentISPaccessHours: 15
excellentISPprice: 19.95
excellentISPextraHoursPrice: 1.00
excellentISPpackage: Standard
```

Here is an example entry for an Excellent ISP customer who has the premium class of service:

```
dn: cn=Mary Chen, o=Excellent ISP, c=US
oc: inetOrgPerson
oc: excellentISPuser
cn: Mary Chen
sn: Chen
excellentISPmailQuotaMB: 30
excellentISPwebSpaceMB: 30
excellentISPaccessHours: Unlimited
excellentISPprice: 29.95
excellentISPextraHoursPrice: 0
excellentISPpackage: Premium
```

Example Entries with Class-of-Service Templates

To save space and time, the shared information in these entries can be moved into a class-of-service template. The shared information in the entries is then replaced with a new attribute whose value indicates which class-of-service template to use.

The attributes from the class-of-service template are added to the entry on a search.

```
dn: cn=John Smith, o=Excellent ISP, c=US
oc: inetOrgPerson
oc: excellentISPuser
cn: John Smith
sn: Smith
excellentISPpackage: Standard
```

```
dn: cn=Mary Chen, o=Excellent ISP, c=US
oc: inetOrgPerson
oc: excellentISPuser
cn: Mary Chen
sn: Chen
excellentISPpackage: Premium
```

Example Class-of-Service Templates

The Excellent ISP company needs to use two templates. The standard-level template is as follows:

```
set class-of-service standard = {
    object class= excellentISPuser
    cos-attr            = excellentISPpackage
    cos-value           = "Standard"
    attribute-values = {

    (type               = excellentISPmailQuotaMB
    value               = "20"
    disposition         = default),

    (type               = excellentISPwebSpaceMB
    value               = "20"
    disposition         = default),

    (type               = excellentISPaccessHours
    value               = "15"
    disposition         = override),

    (type               = excellentISPprice
    value               = "19.95"
    disposition         = default),

    (type               = excellentISPextraHoursPrice
    value               = "1.00"
    disposition         = default)

    }
};
```

The premium-level template is as follows:

```
set class-of-service premium = {
    object class        = excellentISPuser
    cos-attr            = excellentISPpackage
    cos-value           = "Premium"
    attribute-values = {

    (type               = excellentISPmailQuotaMB
    value               = "30"
    disposition         = default),

    (type               = excellentISPwebSpaceMB
    value               = "30"
    disposition         = default),

    (type               = excellentISPaccessHours
    value               = "Unlimited"
    disposition         = override),

    (type               = excellentISPprice
    value               = "29.95"
    disposition         = default),

    (type               = excellentISPextraHoursPrice
    value               = "0.00"
    disposition         = default)
    }
};
```

Set Up Aliases

An *alias entry* is a directory entry that contains the name of another entry. When you search or browse a directory, you can decide whether to resolve aliases (show the details of the target entry) or to show the details of the alias entry itself.

Aliases are a powerful mechanism where you want one entry to have two names.

However, if you do not use them properly, the result can be inconsistent DITs and degraded performance. Following aliases during operations has an additional cost. To increase performance, use aliases sparingly or not at all.

Also, aliases are generally not expanded when they point to an entry in another DSA.

How Aliases Work

An alias entry has the object class of *alias*. An entries of class *alias* has the mandatory attribute *aliasedObjectName*, which contains the DN of the entry that the alias is pointing to.

Alias entries are leaves of the directory information tree. All other entries are *object entries*. Alias entries can point to object entries or other alias entries and thus provide an alternative name for the corresponding object.

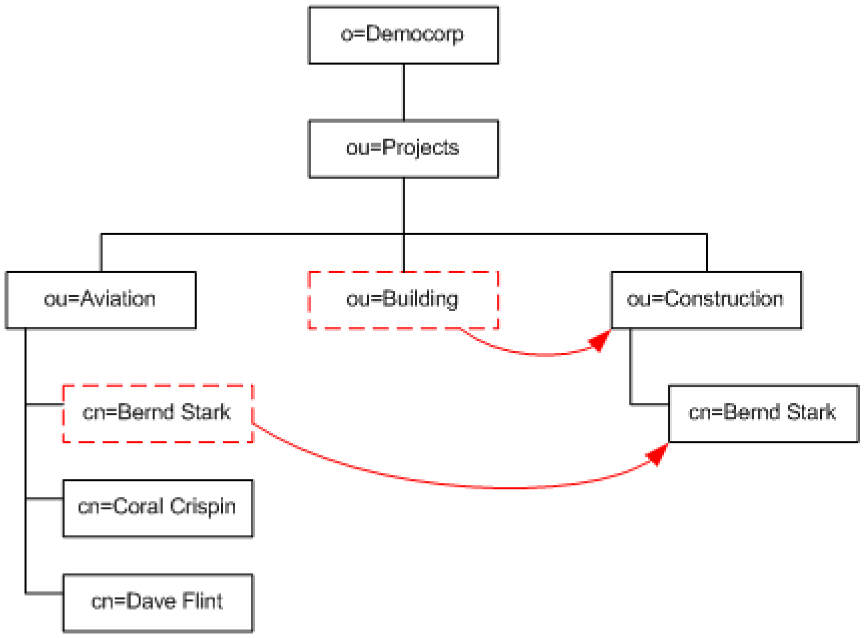
The DSA resolves alias entries to the distinguished name of the object entry to which they point.

For more information, see Name Bindings and Aliases (see page 92).

Example: Aliases in Democorp

The following diagram shows part of the directory information tree for the Democorp DSA. The DIT has been edited to include the following aliases:

- In the Aviation subtree, a new alias entry points to the Bernd Stark entry in the Construction subtree.
- In the Projects subtree, a new alias entry points to the Construction entry.



This example includes the following alias entries:

cn=Bernd Stark,ou=Aviation,ou=Projects,o=Democorp

This is an alias entry that points to *cn=Bernd Stark,ou=Construction,ou=Projects,o=Democorp*.

This entry has the object class *alias*, and the attribute *aliasedObjectName* has the value *cn=Bernd Stark,ou=Construction,ou=Projects,o=Democorp*.

ou=Building,ou=Projects,o=Democorp

This is an alias entry that points to *ou=Construction,ou=Projects,o=Democorp*.

This entry also has the object class *alias*, and the attribute *aliasedObjectName* has the value *ou=Construction,ou=Projects,o=Democorp*.

Alias Integrity

Managing aliases properly is an application design issue.

We recommend that you enable alias integrity. This prevents the creation of aliases that point to no object entry.

More information:

Enable Alias Integrity (see page 395)
Disable Alias Integrity (see page 395)

Enable Alias Integrity

When alias integrity is enabled, the following happens:

- When an entry that has one or more aliases pointing to it is removed, the aliases are also removed.
- Invalid aliases (where no referenced entry exists) cannot be added.

To enable alias integrity, use the following command:

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

Disable Alias Integrity

When you disable alias integrity, the DSA does not prevent the creation of aliases to nonexistent objects. It detects and reports invalid aliases only when it encounters them.

To disable alias integrity, use the following command:

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

Access Controls and Aliases

When a user binds with a user name and password, the system checks the password supplied with the bind against the password in the datastore for the specified user.

By default CA Directory will not allow binds using aliases. To enable aliases to be used on binds, use *set dereference-alias-on-bind = true.*

If you have set CA Directory to allow alias on bind and the user name is an alias, the system checks the password against the password in the entry to which the alias points, but the user name associated with the bind is the alias name.

This means that a user can bind with more than one user name, and each user name can have different access controls associated with it.

How Aliases Affect DIT Searches

When you search a directory information tree that includes aliases, the results depend on the base object, the search scope, and whether aliases are dereferenced during navigating and searching.

The search alias control does not affect the base object of the search, but does determine if an alias entry can be returned in the results of a search. This can have some peculiar effects, especially with base-object searches.

When a directory receives a search request, the directory navigates to the base object that is defined in the request. A base object is the entry from which the search is run. After the directory locates the base object, the search is performed from that location.

If the base object is an alias, the search may request that the alias be dereferenced. An alias entry is *dereferenced* if, during a search, the entry the alias points to is used as the base object rather than the alias itself.

When the directory has located the base object and dereferenced it (if requested), the search is performed. A search may return any number of alias entries. These alias entries may also be dereferenced.

Example: Searching Aliases

This example uses the same DIT as Example: Aliases in Democorp (see page 394).

The base object of a search request is the following alias entry: ou=Building, ou=Projects, o=Democorp. This is an alias that points to ou=Construction, ou=Projects, o=Democorp.

If you choose to dereference aliases while navigating, the searchs run from ou=Construction, ou=Projects, o=Democorp.

However, you can tell the search request to not dereference aliases while navigating. If you do this, the search runs from the alias entry, ou=Building, ou=Projects, o=Democorp.

Example: Searching Aliases Again

This example uses the same DIT as Example: Aliases in Democorp (see page 394).

If you choose to dereference aliases while searching, any aliases returned in the results are dereferenced.

If a search found the alias entry cn=Bernd Stark,ou=Aviation,ou=Projects, the target entry is returned: cn=Bernd Stark,ou=Construction,ou=Projects.

However, you can choose to *not* dereference aliases while searching.

Set Up Dynamic Objects

While dynamic objects can be useful, they go against good DIT design principles. If abused, dynamic objects could lead to major problems.

CA Directory lets you dynamically extend the schema of a DSA in the following ways:

Use ANY attribute defined in the schema (see page 399)

> CA Directory lets you define an object to be *extensible*. The object can then include any attributes that are defined in the schema.
>
> However, we do not recommend this because you cannot control what can be stored against a particular entry. You should be in control of what information is stored and where.

Use ANY attribute NOT defined in the schema (see page 401)

> CA Directory lets you define an object to use *auto-registered attributes*. The object can then dynamically define attributes that are not included in the schema definition.
>
> However, we do not recommend this because you cannot control what can be stored against a particular entry. Also, the attributes cannot be accessed by DXcache, access controls, class-of-service templates, and indexing.

You can extend an object to use auto-registered attributes, any nondefined attributes, or both.

Risks of Dynamic Objects

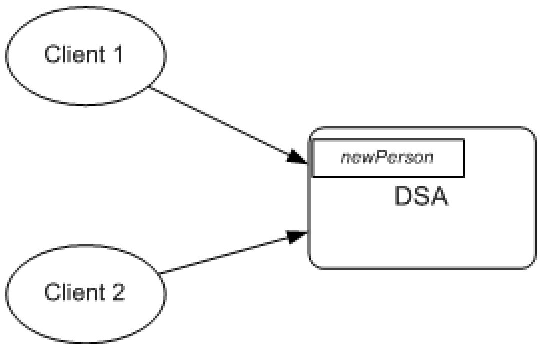
When you define your directory system, you must design your schema carefully. You should try to define your schema without resorting to using auto-registered attributes or extending the object. Do not use dynamic attributes as a way to avoid designing the schema.

If you do use dynamic attributes, you have less control over which attributes can be added to an object. You can still control which of the defined attributes are included, but you cannot control which unknown attributes are added.

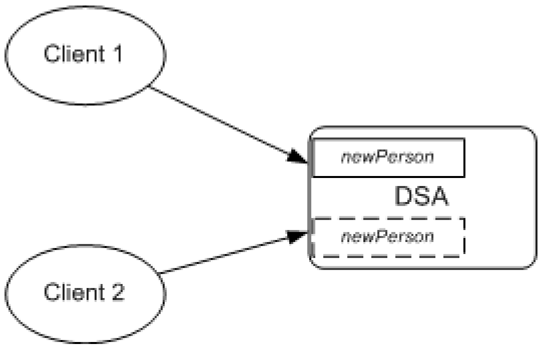
Example: Two Applications Using the Same Auto-registered Attribute

If you use auto-registered attributes, different applications may use an attribute of the same name for different purposes.

For example, the following diagram shows a single DSA that is used by two applications. The application Client 1 uses the auto-registered attribute *newPerson*:



However, problems occur when Client 2 tries to also register an attribute named *newPerson*, which has different parameters, as shown in the following diagram:



Extensible Objects

If you define an object as extensible, it may contain any attribute that *is* already defined in the DSA schema.

Extensible objects are defined in RFC 4512. This standard defines the auxiliary object class *extensibleObject.*

The auxiliary object class *extensibleObject* is defined in the schema file ldapv3.dxc, which is supplied with CA Directory. However, it is not in the default schema group. If you want a DSA to use this schema, you must add it to the DSA's initialization file.

There are two ways to make an object extensible:

- Add the *extensibleObject* object class to an object.

  For instructions, see Make an Existing Object Extensible (see page 400).

- Add *all-attributes* to the may-contain list in the schema.

  For instructions, see Make an Existing Object Class Extensible (see page 399).

Make an Existing Object Class Extensible

If you are writing a schema for a CA Directory DSA, you can make an object class extensible.

To make an existing object class extensible

1. Open the schema file and find the object class that you want to update.
2. Add *all-attributes* to the *may-contain* list, as follows:

   ```
   schema set object-class prefix:2 = {
   ...
   may-contain
       all-attributes
   ...
   };
   ```

3. Initialize the DSA.

Make an Existing Object Extensible

If you are using an existing schema, you can still make an object extensible.

This section describes how to do this using JXplorer.

To make an existing object extensible

1. If the DSA does not include ldapv3.dxc in its schema, do the following:
    a. Add *ldapv3.dxc* to the DSA initialization file after the *clear schemas* command, and after the X.500 schema is sourced, as in the following example:

    ```
    clear schema;
    source "../schema/dxmanager.dxg";
    source "../schema/ldapv3.dxc";
    ```

    b. Initialize the DSA using the following command:

    ```
    dxserver init dsa-name
    ```

2. In JXplorer, connect to the DSA.
3. Navigate to the object that you want to change.
4. In the right pane, select the Table Editor tab to display the attributes for this object.
5. Click the Change Class button.
6. In the Set Object Entry Classes dialog, find the object class *extensibleObject* in the Available Classes list.
7. Select the object class *extensibleObject,* click Add, and click OK.
8. Click Submit to save your change to this object.

    This object can now use any attribute that is defined in the DSA schema.

Add an Attribute to an Extensible Object

After you have made a object extensible, you can add any attribute in the schema to that object.

You cannot do this using JXweb. This section describes how to do this using DSA console commands.

To add an attribute to an extensible object

1. Ensure that the object is extensible, as described in the previous section.
2. Open a DSA console, and bind to the DSA using the following command:

   ```
   bind-req;
   ```

3. Add the attribute to the entry using the following command:

   ```
   mod-entry-req entry=DN add-attr {attribute-name "attribute-value"};
   ```

   For example, to add the *carLicense* attribute with the value EXT 133 to the Democorp organization entry, use the following command:

   ```
   mod-entry-req entry=<c au><o democorp> add-attr {carLicense "EXT 133"};
   ```

Example: Extending an Object Class Definition

The following schema definition defines the object class *newPerson*, which inherits from the *person* object class.

The *person* object class must contain the attributes *cn* and *surname*, which means that the *newPerson* object class must also include those attributes. In addition, it may contain any other defined attribute.

```
schema set object-class myprefix:1 = {
    name = newPerson
    subclass-of person
    may-contain
        all-attributes
    };
```

Auto-Registered Attributes

If you permit an object to use auto-registered attributes, it may contain any attribute that *is not* already defined in the DSA schema.

This can be useful if an LDAP application uses attributes that are not defined in the DSA's schema.

How Auto-Registered Attributes Work with Other CA Directory Features

Auto-registered attributes are supported with LDAP clients only.

The following list describes how auto-registered attributes work with CA Directory tools and features:

DAP Tools

Auto-registered attributes do not work with the DAP tools (DXmodify, DXrename, DXdelete, and DXsearch).

DXloaddb Tool

You can use the DXloaddb tool to load auto-registered attributes from an LDIF file into a datastore. The DXloaddb tool auto-registers an attribute only if it is using the schema from the DSA configuration.

Replication

Auto-registered attributes work with DISP, multiwrite, and multiwrite-DISP DSAs, but only with other CA Directory DSAs.

Other features

Auto-registered attribute names are not supported within the configuration files. This means that special indexing, access controls, class of service, and other features do not support auto-registered attribute names.

Limitations of Auto-Registered Attributes

Auto-registered attributes are limited to 40 characters, and are always multi-valued.

You cannot use an auto-registered attribute as the naming attribute for an object.

Attributes that are automatically registered use the following template for their virtual schema definition:

```
attribute auto-generated-OID = {
    name= supplied-name
    equality    = caseIgnoreMatch
    substr      = caseIgnoreSubstringsMatch
    syntax      = directoryString
}
```

You can use JXweb to display extensible attributes, but you cannot use it to add a new value to an extensible attribute.

Use Auto-Registered Attributes in an Object

To allow a DSA to use attributes from an LDAP client, even if those attributes are not defined in the DSA schema, you need to enable auto-registered attributes in a schema used by the DSA.

To allow an object class to use auto-registered attributes

1. Open the schema file and find the object class that you want to update.
2. Add *auto-register-attributes* to the *may-contain* list, as follows:

```
schema set object-class prefix:2 = {
...
may-contain
    auto-register-attributes
...
};
```

3. Initialize the DSA.

Example: Using Auto-Registered Attributes

The following schema definition defines the object class *autoOrganization*, which inherits from the *organization* object class.

The organization object class must contain the attribute *organizationName* and may contain any attribute in the *organizationalAttributeSet*. These requirements are inherited by the object class *autoOrganization*. In addition, it must contain the attribute *description* and may contain any other attribute that is not defined in the DSA's schema.

```
schema set object-class myprefix:2 = {
    name = autoOrganization
    subclass-of organization
    may-contain
        auto-register-attributes
    must-contain
        description
    };
```

Use Both Auto-Registered Attributes and Extensible Attributes

You can allow an object class to use auto-registered attributes and also any attributes defined in the DSA schema. This allows the class to use all attributes already defined in the schema, and also to contain any attribute that *is not* already defined in the DSA schema.

To allow an object class to use all defined and non-defined attributes

1. Open the schema file, and find the object class that you want to update.
2. Add both *all-attributes* and *auto-register-attributes* to the *may-contain* list, as follows:

   ```
   schema set object-class prefix:2 = {
   ...
   may-contain
       auto-register-attributes
       all-attributes
   ...
   };
   ```
3. Initialize the DSA.

Replicate Password Policy Attributes to Another LDAP Directory

If your CA Directory DSA uses password policies and it is replicated to another LDAP directory, you can replicate some password policy attributes to the other LDAP directory.

To replicate password policy attributes to another LDAP directory

1. Decide which password policy attributes are to be replicated to the other LDAP directory.
2. For each attribute to be replicated, add the following line to the attribute definition in the dxserver.dxc schema file:

   ```
   ldap-names = attribute-name
   ```

   Note: Any password policy attributes that are not marked with *ldap-names* are not included in the replicated update. If no attributes are included, then the update is not sent.
3. Set the following command to *true*:

   ```
   set password-netscape-op-attrs = true | false;
   ```
4. Initialize the CA Directory DSAs.

Example: Replicate Two Password Attributes to a SunONE Directory

In this example, your directory backbone includes a SunONE directory. This is kept synchronized by multiwrite replication between the CA Directory DSA and the SunONE directory.

Your SunONE directory uses the following password attributes:

SunONE Attribute	Equivalent CA Directory Attribute
nsAccountLock	dxPwdLocked
passwordRetryCount	dxPwdFailedAttempts

To include these attributes in the replication, do the following:

1. In the *dxserver.dxc* schema file, find the definitions for the two CA Directory attributes.
2. Add the lines shown in bold to these attribute definitions:

   ```
   schema set attribute dxserver-attr:11 = {
    name = dxPwdFailedAttempts
    ldap-names = passwordRetryCount
    syntax = integer
    single-valued
    no-user-modification
   };
   ```

   ```
   schema set attribute dxserver-attr:14 = {
    name = dxPwdLocked
    ldap-names = nsAccountLock
    syntax = boolean
   };
   ```

3. Add the following command to a .dxc file used by the CA Directory DSA:

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

4. Initialize the CA Directory DSA.

Sample DSML Server

DSML is an XML protocol that permits directory structural information to be represented in XML. The DSML protocol is an almost direct mapping of LDAP. The purpose of the language is to allow XML-based applications to use directory information.

The *DSML Server* lets client applications use DSML, rather than LDAP, to communicate with CA Directory.

DSML can be used anywhere LDAP is used. However, it is usually best to use LDAP. This is because LDAP is the more established protocol and slightly more efficient.

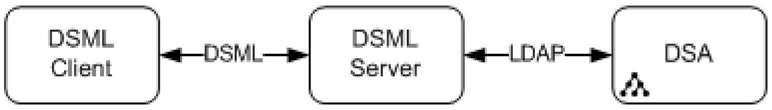
The URL for the DSML Server is as follows:

```
http://localhost:8080/dsml/services/DSML
```

How the DSML Server Works

The following happens when a DSML client uses DSML to communicate with a DSA through the DSML Server:

1. The DSML client sends a DSML request to the DSML Server.
2. The DSML Server translates the DSML request into LDAP and passes it on to the DSA.
3. The DSA responds using LDAP.
4. The DSML Server translates the DSA response from LDAP to DSML and sends it back to the DSML client, as shown in the following diagram:



Server DN and Base DN in the DSML Server

When you connect to the DSML Server, you can specify the server DN. This is the root DN to which you are connected.

For example, if you connect to the Democorp DSA using DSML and you set the server DN to *c=AU*, *c=AU* is not displayed in the tree, and the top node is *o=DEMOCORP*.

If you use JXweb to connect to the DSML Server, you can also specify the base DN. This is the DN of the entry in the tree that you want displayed. The base DN is relative to the server DN. The base DN is not set by the DSML Server: instead, it is set by the client that connects to the server.

For example, if you connect to the Democorp DSA using DSML and you set the server DN to *c=AU*, you could then also set the base DN to *ou=National,ou=Customer,o=DEMOCORP*. Because the base DN is relative to the server DN, you could not set the base DN to *ou=National,ou=Customer,o=DEMOCORP,c=AU*.

How the DSML Server Determines the LDAP URL

To connect to an LDAP server, the DSML Server has to work out the LDAP URL of the DSA.

The DSML Server does this in the following way:

1. The DSML Server checks the POST parameters of the HTTP request: *ldapHost* and *ldapPort*.
2. If these parameters are not supplied, the DSML Server checks the configuration file *dsml.properties* for a server URL.

   For example, the server URL could look like this:

   ```
   ldap://computer27:19289
   ```
3. If the configuration file is not available, the DSML Server uses the default URL.
   - If the DSML Server *can* determine the name of the local machine, this default URL is as follows:

     ```
     ldap://hostname:19289
     ```
   - If the DSML Server *cannot* determine the name of the local machine, this default URL is as follows:

     ```
     ldap://localhost:19289
     ```

Connect to the DSML Server

To connect to a DSML Server

Connect to the server using the following URL:

```
dsml/services/DSML
```

If you use a URL of this form, the DSML client looks for connection details in the *dsml.properties* file.

To connect to a DSML Server on a particular host and port

Connect to the server using the following URL:

```
dsml/services/DSML?ldapHost=computername&ldapPort=port-number
```

In this URL:

computername

Specifies the name of the computer on which the DSML service is running.

port-number

Specifies the port number of the DSML service.

To connect to a DSML Server on a particular host and port with a server DN

Connect to the server using the following URL:

```
dsml/services/DSML?ldapHost=computername&ldapPort=port-number&ldapDN=DN
```

In this URL:

computername

Specifies the name of the computer on which the DSML service is running.

port-number

Specifies the port number of the DSML service.

DN

Specifies the server DN as follows:

%3d

Represents the equal-to character (=).

%2c

Represents the comma character (,).

For example, use the following to represent *o=DEMOCORP,c=AU*:

```
o%3dDEMOCORP%2cc%3dAU
```

Change the DSML Properties in the DSML Server

The DSML properties are stored in a text file. At present, the only property you can control with this file is the URL for the LDAP server.

To change the DSML properties

1. Open the following file in a text editor: DXHOME/../dxwebserver/webapps/dsml/WEB-INF/dsml.properties.
2. Change the URL, and then save the properties file.

Sample Tools

The sample tools are some extra tools for managing CA Directory.

These tools are provided for testing and demonstration purposes. The tools are installed into the following subdirectories in DXHOME/samples. For more information, look in the Readme for each tool.

democorp

A setup script that automatically configures the Democorp DSA and populates it with data for the staff directory for the fictional company, Democorp.

DXsoak

An executable to measure the throughput (in searches per second) of a DSA.

dua

A sample text-based Directory User Agent (DUA) that runs over DAP which you can use to execute scripts to add, modify, or remove entries from the directory. This can also be used to perform searches.

languages

Language certification sample files. These language files have been provided so that the user can test CA Directory operating systems other than English.

Use only the language files that relate to your operating system.

ldua

A LDAP directory client scripting DUA that you can use to execute scripts to add, modify, or remove entries from the directory. This can also be used to perform searches.

snmp

A management client that retrieves management information from a CA Directory DSA using the SNMP protocol.

ssl

Examples of using DUA-DSA and DSA-DSA SSL authentication and encryption.

test

A selection of scripts to modify the directory using the supplied DUA or LDUA clients. The DAP and LDAP script DUAs connect to the Democorp DXserver using SSL encryption or SSL authentication.

trap

A management application that receives SNMP traps from a CA Directory DSA.

unspsc

A setup script that automatically configures the UNSPSC DSA and populates it with data. This data is the United Nations Standard Product and Services Classification (UNSPSC) Code System.

Glossary

alarms

Alarms are reports of critical events that should be monitored.

alerts

Alerts are any events that the user should be made aware of. In this version of DXmanager, the alerts are harvested from the alarm logs.

alias entries

An *alias entry* is a directory entry that contains the name of another entry. When you search or browse a directory, you can decide whether to resolve aliases (show the details of the target entry) or to show the details of the alias entry itself.

association

association is a synonym for binding.

attributes

An *attribute* is a property of an entry that can have a value. An entry is defined by the values of its attributes. For example, an entry in a staff directory could include an attribute named *phoneNumber*, with the value *555-1234-567*.

authentication levels

Each DSA has one or more *authentication levels*. The authentication levels assigned to a DSA define what credentials a user must present to bind to and query that DSA.

auto-registered attributes

An *auto-registered attribute* is an attribute that is used in a directory without being defined in the directory's schema. To use auto-registered attributes in a particular object class, that class must include the keyword *auto-register-attributes* in its *may-contain* list.

auxiliary object class

An *auxiliary object class* defines additional characteristics of an entry. For example, it can provide additional, optional, attributes for an entry.

backbone

The *backbone* is the term used in DXmanager to refer to the directory installation as a whole. It includes all the DSAs and their configuration data.

base-object searches

A *base-object search* specifies a search that returns one node, the base object, as specified by the DN. It is one of the three types of LDAP search. The others are single-level and subtree.

binding

When a DSA, an LDAP client, or a DUA successfully binds to a DSA, the resulting relationship is called a *binding.* Because each binding corresponds to a user, the term user can be used to mean a binding.

class-of-service templates

A *class-of-service template* stores information that can then be included in many entries. Class-of-service templates can reduce the size of a directory, help keep data consistent, and reduce the time required for bulk updates.

credentials

A user's *credentials* are information that identifies them, which are used for authorization. Credentials can be a user name and password, or a certificate.

data DSA

A *data DSA* holds data, and queries are routed to it by router DSAs.

datastore

Each data DSA holds its directory data in memory. The *datastore* is a file that is mapped to the memory image and provides persistent storage of the data.

deploying

Deploying is the process of transferring the XML configuration file from DXmanager to the DSA's host, and then reinitializing the DSAs.

dereferencing

An alias entry is *dereferenced* if, during a search, the entry the alias points to is used as the base object rather than the alias itself.

DIB (directory information base)

The *directory information base* (DIB) is the collection of information held by the directory as a whole (typically in many DSAs).

directory management server

The *directory management server* is the computer on which you have installed the Directory Management package, which includes DXmanager.

DISP

*DISP (*Directory Information Shadowing Protocol) is defined in the 1993 X.525 standard. DISP lets you replicate information in OSI-conformant directories, which permits copying of directory information from one DSA to another using a standardized procedure and protocol.

distribution

In a *distributed* directory, many DSAs cooperate to form a single namespace. Parts of the namespace are served by different DSAs. An application connected to any DSA can search the entire namespace.

DIT (directory information tree)

The *directory information tree* (DIT) is data represented in a hierarchical tree structure, where each node in the tree is defined by a DN. CA Directory uses the term namespace to refer to the directory DIT.

DN (distinguished name)

A *distinguished name* (DN) uniquely identifies a directory entry and its location in the directory namespace. The DN includes the name of the entry, plus the names of all superior entries, for example, *cn=Craig LINK,ou=Administration,ou=Corporate,o=DEMOCORP,c=AU*.

DSA (directory system agent)

A *DSA* is a process that manages some or all of a directory's namespace.

DSA console

The *DSA console* lets you connect to a DSA to give DXserver commands, receive trace information, and act as a user agent.

DSML (Directory Services Markup Language)

DSML is an XML protocol that permits directory structural information to be represented in XML. The DSML protocol is an almost direct mapping of LDAP. The purpose of the language is to allow XML-based applications to use directory information.

DSML Server

The *DSML Server* lets client applications use DSML, rather than LDAP, to communicate with CA Directory.

DSP (Directory System Protocol)

The *directory system protocol* (DSP) is a protocol used between DSAs for X.500 distributed operations. This is the protocol used for chaining. LDAP does not have a DSP equivalent. A CA Directory DSA can use both DSP and LDAP.

DUA (directory user agent)

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.

DXadmind

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.

DXmanager

DXmanager is a web application that lets you create, configure, monitor, and control your directory backbone.

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.

dynamic groups

A *dynamic group* is an entry in the directory with its membership defined by an LDAP filter. All entries that satisfy this filter are members of the dynamic group.

EIS (entry information selection)

In a search filter, the *entry information selection* (EIS) is the attributes that are to be returned in the search results.

entries

The *entry* is the basic unit of information storage in a directory. All information in a directory is stored in the form of entries, which are also sometimes named *objects*. For example, in a directory listing all staff members at a company, each staff member would have an entry.

extensible objects

An *extensible object* is an object class that may include any attribute defined in the DSA schema.

failback

Failback is the restoration to normal service of a CA Directory DSA after failing over and recovering.

failover

Failover is the ability of a router DSA to continue to service queries even when a data DSA becomes unavailable. If the router detects that a DSA has failed, it resends outstanding requests to another DSA that serves the same partition, making the failure invisible to clients.

health

A directory's *health* is a measure of whether the directory is running smoothly. This includes whether all DSAs are running, and whether any are logging warning, error, or alarm messages.

horizontal partitioning

If you have a large flat namespace, you can improve performance by partitioning the namespace, so that different DSAs each serve different parts of the same level of namespace. In CA Directory, *horizontal partitioning* is a method of doing this.

hosts

A *host* is a single computer with CA Directory installed on it. A single host may serve one or more namespace partitions.

hub

A *hub* is the DSA that is responsible for receiving multiwrite update requests from other regions.

idle time

The *idle time* for a connection is the time elapsed since the DSA last performed an operation on that connection.

instantiation

In DXmanager, the process of assigning hosts, sites, or regions to namespace partitions is called *instantiation.* After the configuration is deployed, this process creates one or more DSAs to serve the namespace partitions.

JNDI (Java Naming and Directory Interface)

The *Java Naming and Directory Interface* (JNDI) is a Java API that provides applications based on Java with naming and directory services.

JXweb

JXweb is a general purpose LDAP-compliant directory browser and editor, which lets you access a directory through the Web.

latency

Latency is the delay caused by the round-trip time taken between sending a network packet and receiving a response. For replication, latency describes the time during which the shadow servers are out-of-date with respect to a master.

LDAP (Lightweight Directory Access Protocol)

LDAP is a protocol for accessing directories. LDAP is a simplified version of the X.500 directory access protocol (DAP).

LDIF (LDAP Data Interchange Format)

LDIF is a format suitable for describing directory information or changes to be made to directory information.

LDIF files

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.

LDT file

An *LDT file* is a text file that contains rules for transforming data into LDIF format. The csv2ldif tool uses an LDT file to transform CSV data into LDIF.

leaf entries

A *leaf entry* is a directory entry that has no subordinate entries.

load

A directory's *load* is a measure of the amount and type of operations being sent to that directory.

load sharing

Load sharing lets a router DSA distribute incoming requests evenly among all DSAs in the same site that serve the same namespace partition. This improves performance.

logs

The *logs* contain output from a DSA, including its trace, diagnostics, warnings, and alarms.

MIB (management information base)

A *management information base* (MIB) is a database of objects that can be managed using SNMP.

multiwrite group

A *multiwrite group* is a synonym for a region.

multiwrite peers

Multiwrite peer DSAs are the DSAs in a multiwrite replication system. Multiwrite peers service the same prefix, share knowledge of each other, and their knowledge includes the DSA flag *multi-write*.

multiwrite replication

Multiwrite replication is a mechanism for replicating updates to a number of DSAs to ensure that they are synchronized. When a DSA receives an update, it updates its own data and then sends the update to its peers. If a peer DSA cannot be reached, the updates are queued and replayed when the DSA becomes available.

multiwrite-DISP replication

Multiwrite-DISP replication is a replication scheme that uses multiwrite replication for real-time updates and DISP for recovery.

namespace

A *namespace* is the tree of all data in the directory, synonymous with directory information tree. It is defined by the DN of the top node in the tree.

namespace partition

A *namespace partition* is a sub-section of the directory information tree.

naming attributes

The *naming attribute* is the attribute used to form the RDN, which uniquely identifies each entry in the directory.

network topology

The *network topology* describes the hosts on which the directory backbone runs, and the quality of network connections between the hosts. The quality of connections is represented by *sites* and *regions*.

OID (object identifier)

An *object identifier* (OID) is a numeric value that unambiguously identifies an object class, attribute, or syntax in a directory service. An OID is represented as a dotted decimal string (for example, *1.2.3.4*). Companies and individuals can obtain a root OID from an issuing authority and use it to allocate additional OIDs.

OID prefix

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

operational attributes

An *operational attribute* represents information used to control the operation of the directory (such as access control information), or used by the directory to represent some aspects of its operation.

operations

A directory *operation* reads data from the directory or writes data to it. Operations include add, compare, delete, modify entry, modify RN, read, search.

performance

Directory *performance* measures the load and throughput of the directory. DXmanager can tell you which DSAs and hosts are under the most load, and the kinds of operations they are performing (searches, updates, compares, and so on).

personality certificates

DSA *personality certificates* are the same as user certificates, except that they are for DSAs. These personality certificates permit the links between DSAs to be secured using SSL encryption or authentication.

phase

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.

public user

A *public user* is a user who has not been authenticated. The following terms are identical: public user, unauthenticated user, anonymous user, user who has not logged in.

RDN (relative distinguished name)

The *relative distinguished name* (RDN) of an entry is the lowest-level part of the entry's DN. The RDN is formed by the entry's naming attribute. For example, if an entry's DN is *cn=Craig LINK,ou=Administration,ou=Corporate,o=DEMOCORP,c=AU*, the entry's RDN is *cn=Craig LINK*.

recovery

Recovery is the process of a DSA returning to service after an outage.

recovery mode

A DSA in *recovery* mode is one that has been offline, so its data may now inconsistent with its replication peer DSAs. In recovery mode, a DSA only accepts binds and updates from its replication peers. This prevents clients and parent or non-peer DSAs from querying or updating the recovering DSA.

recovery notification list

A *recovery notification list* is a list of peer DSAs that have more up-to-date data. When the data in a recovering DSA has been synchronized with one of these peers, the name of the peer is removed from the list. When the list is empty, the DSA is recovered and returns to service.

referential integrity

A directory entry has *referential integrity* if all DNs in the entry are valid, that is, point to other, existing, entries.

region

A *region* is a collection of sites. Multiwrite replication between DSAs in the same region is synchronous.

response files

A *response file* is a text file that supplies information used during the installation process. The user normally supplies this information during the installation process.

router DSA

A *router DSA* has no local data and no datastore. It can only route traffic to other DSAs.

SAML (Security Assertion Markup Language)

Security Assertion Markup Language (SAML) is an XML-based protocol for making statements about security. For example, you could use SAML to make the following statement: "John Citizen has been authorized to read data from the HR web site for thirty minutes starting at 9:15 a.m."

schema

A *schema* is a formal definition of the contents and structure of the directory data. It governs where each entry can be placed within the directory structure, how entries are to be named, and what attributes each entry can contain.

script files

Script files store frequently used, or complex commands (such as a series of searches). You can execute these files from a DSA console, or by using the source command, from other script files.

selective shadowing

Selective shadowing is the ability to replicate only some information to a shadow DSA.

sites

A *site* is a DXmanager term for collection of hosts that are connected by a reliable, fast and low latency network, such as a LAN. hosts A site corresponds to a load sharing group.

SPML (Services Provisioning Markup Language)

Services Provisioning Markup Language (SPML) is an XML-based protocol that handles provisioning and user management. It allows for adding, modifying, deleting, and searching users, and provisioning users with resources.

static groups

A *static group* is an entry in the directory with a *member* attribute, which stores a list of the DNs of the entries that are members of this group.

TCP port

The *TCP port* is the port on which DSA listens for, and accepts connections from, clients and other DSAs.

text-based configuration files

You can configure CA Directory using commands in text files. The text files contain commands that define how the DSA works. These commands are identical to the commands that can be entered from a DSA console.

third-party LDAP server

A *third-party LDAP server* is a directory that only uses the LDAP standard. DXmanager can monitor these LDAP servers, but it cannot configure or control them.

third-party X.500 DSA

A *third-party X.500 DSA* is a directory served by another directory product. DXmanager can monitor and control these DSAs, but it cannot configure them.

topology

The *topology* defines the network in terms of a hierarchy of *regions*, *sites*, and *hosts*. A group of hosts belongs to a site, and a group of sites belongs to a region. Because the topology is hierarchical, a site cannot contain a region and a host cannot contain a site or region.

traces

A DSA's *trace* is its record of almost all operations going into and out of that DSA. You can view a DSA's trace in the log files, and also by using the DSA console.

transparent routing

Transparent routing allows a router DSA to process LDAP requests and responses without requiring the controlling schema. This is useful if the router DSA is being used to link LDAP clients to an LDAP server, or the clients and server have schema that are not known by the router DSA. Transparent routing works with LDAP clients only.

view

A *view* is a read-only virtual directory that provides a layer between the user agent and the real directories. It lets you combine multiple LDAP searches into one search. You can use views to improve performance and to reduce the complexity of applications.

X.500

X.500 is a set of computer networking standards that define directory services. The protocols defined by the X.500 standards include DAP, DSP, and DISP. A directory that follows the X.500 standard has distributed operations, distributed management, distributed security, and replication.