Sun Directory Services 3.1 Administration Guide

Sun Microsystems, Inc.
901 N. San Antonio Road
Palo Alto, CA 94303-4900
U.S.A.

Part No: 805-6061
June 1998

Contents

Preface

Sun™ Directory Services offers a global directory and naming service. The product contains:

- A Lightweight Directory Access Protocol (LDAP) server, compliant with the LDAP v3 Internet standards
- An LDAP replication service
- A Network Information System (NIS) server, that integrates into an existing NIS environment to provide an integrated naming service
- A Remote Access Dialup User Service (RADIUS) server that provides authentication for remote users connecting to the network through a Network Access Server (NAS), and also provides accounting services
- A Java™ administration console that offers local and remote configuration and management
- A Java directory editor, called Deja, for maintaining directory information
- A web gateway that offers access to the directory from any web browser
- A set of administration commands, including data import and export utilities
- SNMP agents for monitoring by a management application

The Sun Directory Services also offer the following security features:

- Optionally, encryption of communications between client and server using SSL
- Strong authentication in bind through the SASL protocol
- Password encryption in RADIUS transactions

The *Sun Directory Services 3.1 Administration Guide* describes the configuration and management tasks that need to be performed to maintain directory information and operate a directory service. It also describes how to integrate the NIS server provided as part of the Sun Directory Services into a NIS environment.

This book is intended for directory server administrators.

How This Book Is Organized

Sun Directory Services 3.1 Administration Guide is organized as follows:

Chapter 1 describes the concepts governing a global and distributed directory service.

Chapter 2 briefly describes the overall architecture and the individual components of the Sun Directory Services.

Chapter 3 provides guidelines for organizing the structure of the directory according to the type of information you want to store in it.

Chapter 4 describes the default configuration of Sun Directory Services and the configuration tasks that you can perform to tailor the configuration to meet your specific needs.

Chapter 5 describes how to populate and maintain the database.

Chapter 6 explains how to use the directory server to replace an NIS server. It contains configuration information for the NIS service.

Chapter 7 explains how to use the directory to authenticate users through the RADIUS protocol. It contains configuration information for the RADIUS service.

Chapter 8 explains how to modify the default schema, and also provides an object class and attribute reference for the default schema.

Chapter 9 explains how to set up a replication plan for distributing directory information.

Chapter 10 describes how to start and stop the directory services, and how to gather statistics and accounting information from the SNMP agents supplied with Sun Directory Services.

Related Books

The document set for Sun Directory Services 3.1 also contains:

- *Sun Directory Services 3.1 User's Guide*

Standards Reference

Sun Directory Services is based on the following Internet standards and proposed standards:

- *RFC 1777 Lightweight Directory Access Protocol*
- *RFC 1778 The String Representation of Standard Attribute Syntaxes*
- *RFC 2138 Remote Authentication Dial In User Service*
- *RFC 2139 RADIUS Accounting*
- *RFC 1565 Network Services Monitoring MIB*
- *RFC 1567 X.500 Directory Monitoring MIB*
- *RFC 2222 Simple Authentication and Security Layer*
- *RFC 2251 Lightweight Directory Access Protocol (v3)*
- *RFC 2252 Lightweight Directory Access Protocol (v3) Attribute Syntax Definitions*
- *RFC 2253 Lightweight Directory Access Protocol (v3) UTF-8 String Representation of Distinguished Names*
- *RFC 2254 The String Representation of LDAP Search Filters*
- *RFC 2255 The LDAP URL Format*
- *RFC 2256 A Summary of the X.500(96) User Schema for Use with LDAPv3*
- *RFC 1274 The COSINE and Internet X.500 Schema*
- *draft-byrne-alias-00.txt*
- *draft-howard-nis-schema-04.txt*
- *draft-ietf-asid-ldapv3-referral-00.txt*
- *draft-ietf-asid-ldapv3-tls-02.txt*
- *draft-ietf-asid-ldapv3-lang-02.txt*
- *draft-ietf-asid-ldapv3-ext-04.txt*
- *draft-ietf-asid-ldif-02.txt*
- *draft-ietf-radius-auth-servmib-01.txt*
- *draft-ietf-radius-acc-servmib-01.txt*
- *draft-ryan-java-00.txt*

Sun Directory Services supports the following OSI standards:

- *ITU-T X.500 1988 Recommendations*
- *ITU-T X.208 Abstract Syntax Notation One (ASN.1)*
- *ITU-T X.209 Basic Encoding rules (BER)*

Ordering Sun Documents

The SunDocs^{SM} program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals using this program.

For a list of documents and how to order them, see the catalog section of the SunExpress^{TM} Internet site at `http://www.sun.com/sunexpress`.

Accessing Sun Documentation Online

The `docs.sun.com` Web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com`archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`.

What Typographic Changes Mean

The following table describes the typographic changes used in this book.

TABLE P–1 Typographic Conventions

Typeface or Symbol	Meaning	Example
`AaBbCc123`	The names of commands, files, and directories; on-screen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% You have mail.`

`AaBbCc123`	What you type, contrasted with on-screen computer output	`machine_name%` **`su`**
		`Password:`

TABLE P–1 Typographic Conventions *(continued)*

Typeface or Symbol	**Meaning**	**Example**
AaBbCc123	Command-line placeholder: replace with a real name or value	To delete a file, type `rm` *filename*.
AaBbCc123	Book titles, new words or terms, or words to be emphasized	Read Chapter 6 in *User's Guide*. These are called *class* options. You *must* be root to do this.

Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

TABLE P–2 Shell Prompts

Shell	**Prompt**
C shell prompt	`machine_name%`
C shell superuser prompt	`machine_name#`
Bourne shell and Korn shell prompt	`$`
Bourne shell and Korn shell superuser prompt	`#`

CHAPTER **1**

Introduction to Directory Concepts

This section provides information on:

- Directory information
- Directory structure
- Infrastructure information
- Access control
- Replication

Directory Information

The information held in the directory consists of entries and alias entries, which provide the information base, and of infrastructure information, which determines how the information base is structured.

Directory Entries

A *directory entry* is a set of *attributes* and their *values*. Every entry has an *object class* attribute, which specifies the kind of object the entry describes and defines the set of attributes it contains. For a given object class, some attributes are mandatory and some are optional. For example, the object class country is described by the mandatory attribute countryName, and the optional attributes description and searchGuide.

The schema defines the attributes that are mandatory and optional for an entry of a given object class. It also defines an inheritance hierarchy. All object classes inherit the characteristics of their superior object class. For example, the object class organizationalPerson is a subclass of the Person object class. It inherits the mandatory and optional attributes from the Person object class.

All object classes inherit from the object class top. The object class top contains the mandatory attribute objectClass. This ensures that all entries have an object class attribute.

There are two types of object classes: structural and auxiliary. A *structural* object class defines the type of entry. An entry can have only one structural object class. An *auxiliary* object class is not sufficient to define alone a type of entry, but can be specified in addition to a structural object class. For example, the auxiliary object class uidObject can be used to assign a uid to any entry in the directory.

Distinguished Name and Relative Distinguished Name

Directory information is hierarchical, with entries organized in a tree structure. Each entry has a parent entry and can have child entries. The top of the hierarchy is known as the *root entry*.

An entry is identified by its *distinguished name* (DN). A distinguished name is a sequence of attributes and values. The first attribute and its value provide the entry's *relative distinguished name* (RDN). The rest of the sequence is the distinguished name of the parent entry. A distinguished name is unique throughout the whole directory service.

Figure 1–1 shows an example of how directory information is structured, with the DNs and RDNs of the shaded entries.

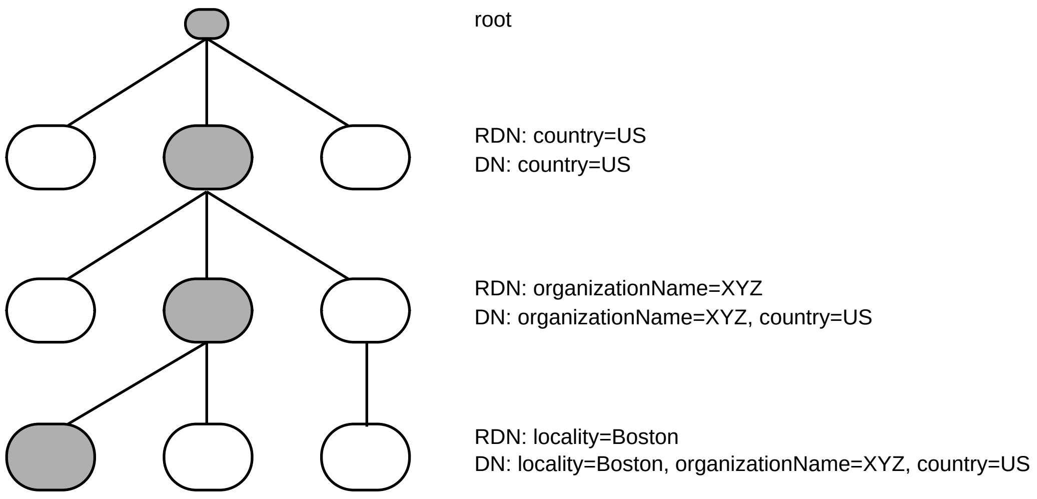

root
RDN: country=US
DN: country=US
RDN: organizationName=XYZ
DN: organizationName=XYZ, country=US
RDN: locality=Boston
DN: locality=Boston, organizationName=XYZ, country=US

Figure 1–1 Directory Information Structure

The attributes that can be part of a DN are called *naming attributes*. In the default schema provided with Sun Directory Services, the naming attributes are:

- countryName, or c
- localityName, or l
- organizationName, or o
- organizationalUnitName, or ou
- commonName, or cn
- channelName, or ch
- domainComponent, or dc
- stateOrProvinceName, or st
- streetAddress, or street
- uid
- ref

Aliasing

You can define an *alias entry*. An alias entry is identified by a distinguished name (DN). It contains the name of the directory entry it represents (the aliased object name). The alias entry and the entry it represents must be under the same root entry. For the full definition of the alias object class, refer to "Object Class Reference" on page 131.

For bind and search operations, you can specify that the directory should translate an alias DN to the DN of the actual entry. This is known as *dereferencing* the alias. For other operations, you need to treat the alias entry as an ordinary entry and not dereference it, for example, when you modify the RDN of the alias entry itself, not of the aliased object.

Alias Entries and Searching

The result of a search or read operation involving an alias entry differs depending on whether or not you dereference the alias. Alias dereferencing is specified by the LDAP client. There are four possible settings for the alias dereference flag:

- Never dereference alias

  All operations apply to the entry with the given DN, even though the entry is an alias entry. This is the default setting.

- Dereference alias when finding base object

  The base object identifies the top of the subtree of entries to be searched. This setting means that if you specify an alias as the base object it will be dereferenced, but no other aliases encountered during the search are dereferenced.

- Dereference alias when searching

  If the operation being carried out is a search, all alias entries specified or used in the search are dereferenced, except the base object. If the result of the search is an alias entry, the aliased object is returned to the user, not the alias entry. This can sometimes lead to unexpected results from searches based on DN content, where the requested information is not present in the entries returned because the entry that contains the requested DN term is an alias entry that has been dereferenced.

- Always dereference alias

  All alias entries specified or used in the operation are dereferenced.

For example, suppose your directory contains the following pair of entries:

```
cn=Stan Smith, role=Personnel Administrator, ou=Personnel, o=XYZ, c=US

with attributes:    objectclass=orgPerson

                    cn=Stan Smith

                    telephoneNumber=123 456 7890

                    mail=dtmail


cn=personnel, o=XYZ, c=US
```

with attributes:	objectclass=alias
	objectclass=aliasObject
	cn=personnel
	aliasedObjectName="cn=Stan Smith, role=Personnel Administrator, ou=Personnel, o=XYZ, c=US"

With alias dereferencing when searching, if you search for the telephone number of cn=personnel in the subtree o=XYZ, c=US, you will see Stan Smith's telephone number. With no alias dereferencing, you would not see any telephone number.

Defining aliases for roles is particularly useful when the person occupying a role changes frequently (the duty network manager for out-of-hours calls, for example), so that users always query the same entry. You can change the value of the aliasedObjectName with a script that runs on a schedule and calls `ldapmodify` to make the changes.

See the `ldapsearch(1)` man page for details of how to specify how alias dereferencing is used in `ldapsearch`.

Alias Entries and Authentication

To perform certain operations on the directory, users need to be authenticated. This is the case for modifications of directory content, or for reading the userPassword attribute of an entry. The level of access permitted is established during the bind process. For details, refer to "Binding to the Directory" on page 9.

The DN supplied in a bind request can be the DN of an alias entry. With alias dereferencing, the user binds with the DN contained in the aliasedObjectName of the alias entry, and is granted the access rights defined for the entry with that DN.

Alias dereferencing during a bind operation is a configuration choice that needs to be made for the LDAP server. If aliases are not being dereferenced and the user binds with the DN of an alias entry, access is denied because the password attribute is not present. This also means that if you allow alias dereferencing, you allow users to bind without a password.

For information on specifying alias dereferencing for Sun Directory Services, refer to "Configuring LDAP Parameters" on page 41.

Directory Structure

In a general-purpose directory, you have to decide what information you want to store, and how that information will be organized. The information in the global directory can be physically located on a number of different servers. However, it fits into a single overall tree structure.

Directory Information Tree

Information in the directory is organized in a tree structure, called the *Directory Information Tree (DIT)*. The structure of a DIT usually reflects very closely the structure of the information it contains. For example, a directory containing entries for people in a corporation could be organized by division or by location. In general, DIT structures are organizational, geographical, or include both organizational and geographical factors. However, another possibility is to organize the DIT structure according to internet domains. The logical organization of the DIT into naming contexts is discussed in Chapter 3.

Data Stores and Naming Contexts

The directory information is divided into naming contexts. A *naming context* is a subtree of the directory, and is identified by the DN of the entry at the top of the subtree. A naming context is stored in a physical *data store*. A data store can hold more than one naming context. A directory server can contain more than one data store.

The naming contexts that identify the data store are called data store suffixes in the Sun Directory Services Admin Console.

When dividing the DIT into naming contexts and data stores to be held on individual servers, you need to take account of the following:

- A search cannot cross naming contexts or data stores. If you want to perform a search in several naming contexts, you have to define referrals between the data stores, or start several simultaneous search operations.
- Alias definitions are local to a naming context.

The Sun Directory Services data store can hold up to four data store suffixes, and one million entries. If you need to store more entries, you can install several data stores, on one or several servers, and create referrals between them. For more information on using Sun Directory Services to store more than one million entries, refer to “Referrals” on page 29.

Infrastructure Information

Infrastructure information determines how the components of a directory service behave and how directory entry information is interpreted. It includes the directory schema, knowledge information, and component configuration information.

Schema

The directory schema is the set of rules that describes the data that can be stored in the directory. It defines the types of entries permitted, their attribute structure, and the syntax of the attributes. Sun Directory Services contains a pre-defined schema, which you can modify, with certain restrictions. See “Modifying the Schema” on page 127 for details of how to modify the schema.

Knowledge Information and Referrals

A directory server uses knowledge information to pass requests for information to other servers. The knowledge information held by a directory server is a reference to a directory server holding other naming contexts. When a server receives a request for information, it checks whether it can respond to the request using the information in the local data store. If it cannot, it checks the referral defined for the data store, and returns the details of an alternate directory server to the directory client. The client can then send the request to the other directory server. Some clients contact the alternate server automatically, so the referral mechanism is transparent to the user. Other clients return the referral information to the user.

See “Example: Naming Contexts in XYZ Corporation” on page 25 for an example of how referrals are used.

Access Control

Access to information in the directory is controlled by a set of rules that determine what operations a user can perform on a particular entry or attribute. The permission level granted to the user depends on the authentication information provided by the user. It also depends on the specific rules defined by the directory administrator for a particular entry or attribute.

Permission Levels

There are five permission levels for directory information. From the least privileged to the most privileged, they are:

- none

  You are not permitted to access the entry at all, and will not even see information indicating that the entry exists.

- compare

  You can compare the value of a given attribute with a value you supply, but you cannot read the attribute value. This is used when checking passwords.

- search

  You can read the distinguished name of an entry, and you can search for entries based on the existence of an attribute or attribute value. You will not necessarily be able to read the attribute value.

- read

  You can read the value of any accessible attribute within an entry.

- write

  You can write information into an entry or attribute, that is, you can modify or delete an attribute value, attribute, or whole entry.

Note - When you are granted permission for a given level of operation, you are implicitly granted all lower levels of permission. For example, read permission implies that search and compare permissions are granted too.

Defining Rules for Entries and Attributes

Access control rules define which users are granted which permission for a given set of entries or attributes. For example, you can give a user read permission for all attributes except password in all entries, and compare permission for password attributes.

You can define the set of entries or attributes to which an access control rule applies by using:

- The symbol * to represent all entries
- A regular expression in a distinguished name (see “Using the DN Editor” on page 55)
- An LDAP filter (see “Using the Filter Editor” on page 57)
- The presence of a particular attribute

For example, you could define the following access control rules:

- Users have write access to their own password attribute, but only compare access to the passwords of other users.
- A user whose entry contains the attribute value locality=San Francisco has read access to all other entries that contain the attribute value locality=San Francisco, but cannot read the password attribute value.

The access control rules are applied in sequence, so the order in which they are listed is important. You must state the most specific rules first, with more general rules afterward. “Configuring Access Control” on page 49 explains how to define an access control rule using the configuration tool, and how to specify the order of rules.

Binding to the Directory

Depending on the access control rules defined for the directory, for certain operations, you need to *bind* to the directory. *Binding* means authenticating yourself by providing your DN and password. This process determines the permission level you are granted for the duration of the connection.

For example, with the default set of access control rules, you have write permission to your password in your directory entry. When you bind with the DN and password of an entry, you are identified by the keyword *self* for that entry. With an anonymous bind, you have search, and read permissions for all entries and attributes except the password attribute, for which you have compare permission. These are the permissions granted to users identified by the keyword *everyone* or *.

Default Access Control Rules

The default access controls defined at installation are as follows:

- All users have compare access to the values of the attribute userPassword. To change the value of the userPassword attribute, you must bind with the DN of the entry containing the attribute, that is, the password can only be changed by the owner of the entry.
- All users have compare access to the values of the attributes chapPasswd, radiusLoginPasswd, radiusPppPasswd, radiusSlipPasswd. Write access to these attributes is granted to the users who bind with the DN of the entry containing the attributes.
- Anyone binding with the DN of an entry has write access to the userPassword, gecos, and loginShell attributes of that entry. Everyone else has read access only.
- Everyone has read access to the following attributes: cn, dataSource, homeDirectory, messageStore, messageStoreSizeQuota, mail, mailServer, objectStatus, preferredRfc822Recipient, rfc822Mailbox, uid.
- Any user can add their DN to, or delete their DN from, the member attribute of any entry containing the attribute joinable with value TRUE.

- The administrator always has complete access to all attributes in all entries. You cannot change the access granted to the administrator because it is not defined in the configuration file. This ensures that there is always at least one user who has access to every attribute in every entry in the directory.

These rules are applied in order, starting with the most specific followed by the more general rules.

Code Example 1–1 shows how the default access controls are defined in the directory server configuration file `/etc/opt/SUNWconn/ldap/current/dsserv.acl.conf`.

CODE EXAMPLE 1–1 Default Access Controls

```
access to attrs=userPassword by self write
 by * compare

# Radius ACLs
access to attrs=chapPassword, radiusLoginPasswd, radiusPppPasswD,
radiusSlipPasswd
 by self write
 by * compare

access to attrs=sharedKey
 by self write
 by * compare

# dsyppasswdd ACLs
access to attrs=userPassword
 by self write
 by * compare

access to attrs=gecos,loginShell
 by self write

# SIMS ACLs
access to attrs=cn, dataSource, homeDirectory, mail, mailHost,
mailQuota, objectStatus, preferredRfc822Recipient, rfc822Mailbox,
uid
 by self read
 by * read

# Default ACLs
access to filter="joinable=TRUE" attrs=member,entry
 by dnattr=member selfwrite

access to * by self read
```

For information on configuring access control rules for Sun Directory Services, refer to “Configuring Access Control” on page 49.

Replication

You can share between several directory servers the load of processing requests generated by directory service clients for the same information. This is done by defining a replica, or slave, server to provide an alternative access point to the directory service for clients. A master naming context can have more than one replica naming context. Figure 1–2 shows a master server with two replica servers. *Replication* is the process by which changes in the master data store are propagated to all the replica naming contexts. You can replicate an entire naming context, a subtree, or a particular entry. You can replicate the entire content of an entry or you can specify a subset of attributes to be replicated.

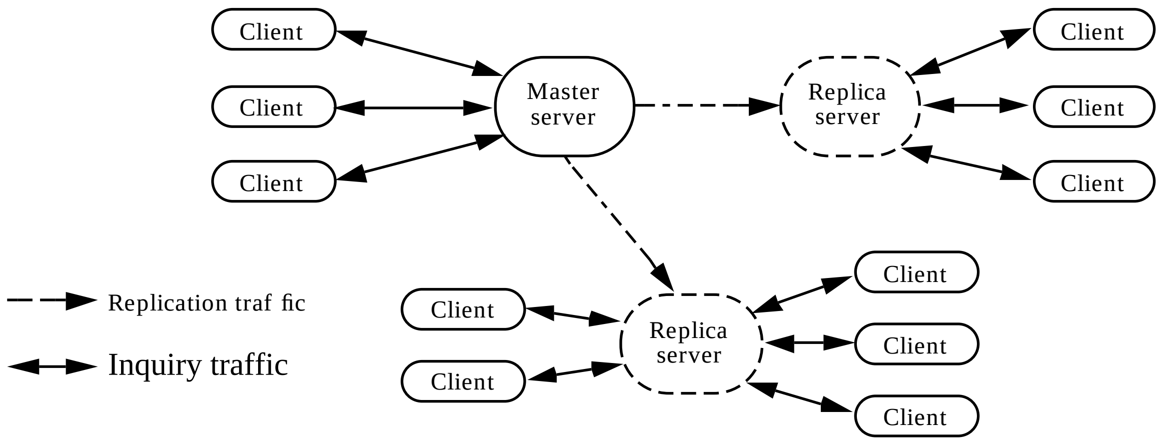


Figure 1–2 Master and Replica Servers

Using replication has the following advantages:

- It reduces the load on the master server by diverting some traffic to other servers.
- You can store copies of data where it is mostly frequently used, reducing network traffic.
- You can store many copies of the data, but the data is maintained from a central location.
- You need only replicate the data that is required by clients of the replica server, if you know the requirements of those clients precisely enough. You may be able to tailor a replica exactly to the needs of a specific client. By reducing the number of entries replicated, you reduce the network traffic caused by replication updates.
- You could maintain a "public" replica server containing information that is not confidential, allowing greater access to this information than you usually allow for other servers. For example, you could create a server containing the email addresses for the sales and support staff who deal with current products but not

the research staff working on future products, and make it available to the sales staff of a partner company.

Note - You could provide the same partial view of directory information with appropriate access controls. However, using a partial replica on a dedicated machine ensures that you are not providing access to your entire network. For extra security, you could connect the replica server to your network only while the replication update is in progress.

The costs of using replication are:

- Additional network traffic caused by replication of data. However, though there may be an overall increase in traffic, more of the traffic will be local, so you can avoid known network bottlenecks for inquiry traffic. Also, you can time the replication updates for when the network is least busy.
- Information retrieved from replicas may be out of date if replication has not happened since an update, so certain applications may always need to query the master data store.
- You cannot modify a replica. All updates must be performed on the master copy of an entry.

CHAPTER **2**

Sun Directory Services Components and Architecture

Sun Directory Services offers a global directory and naming service. The product contains:

- A Lightweight Directory Access Protocol (LDAP) server, compliant with the LDAP v3 Internet standards
- An LDAP replication service
- A Network Information System (NIS) server, that integrates into an existing NIS environment to provide an integrated naming service
- A Remote Access Dialup User Service (RADIUS) server that provides authentication for remote users connecting to the network through a Network Access Server (NAS), and also provides accounting services
- An administration console that offers local and remote configuration and management
- A Java directory editor, called Deja, for maintaining directory information
- A web gateway that offers access to the directory from any web browser
- A set of administration commands, including data import and export utilities
- SNMP agents for monitoring by a management application

The Sun Directory Services also offer the following security features:

- Optionally, encryption of communications between client and server using SSL
- Strong authentication in bind through the SASL protocol
- Password encryption in RADIUS transactions

This chapter provides an overview of the Sun Directory Services architecture and describes the product components. It also provides an overview of security features in Sun Directory Services.

Sun Directory Services Architecture

Sun Directory Services offers access to directory information through a number of different protocols:

- LDAP, for messaging, directory, and management applications
- RADIUS, for authentication applications
- NIS, for all applications that use NIS naming services
- HTTP, for access through any web browser

Sun Directory Services also includes the following Java tools that you can run as applets in any Java-enabled web browser, or as applications:

- An Administration Console for local or remote configuration and administration
- The Deja tool for updating the directory database

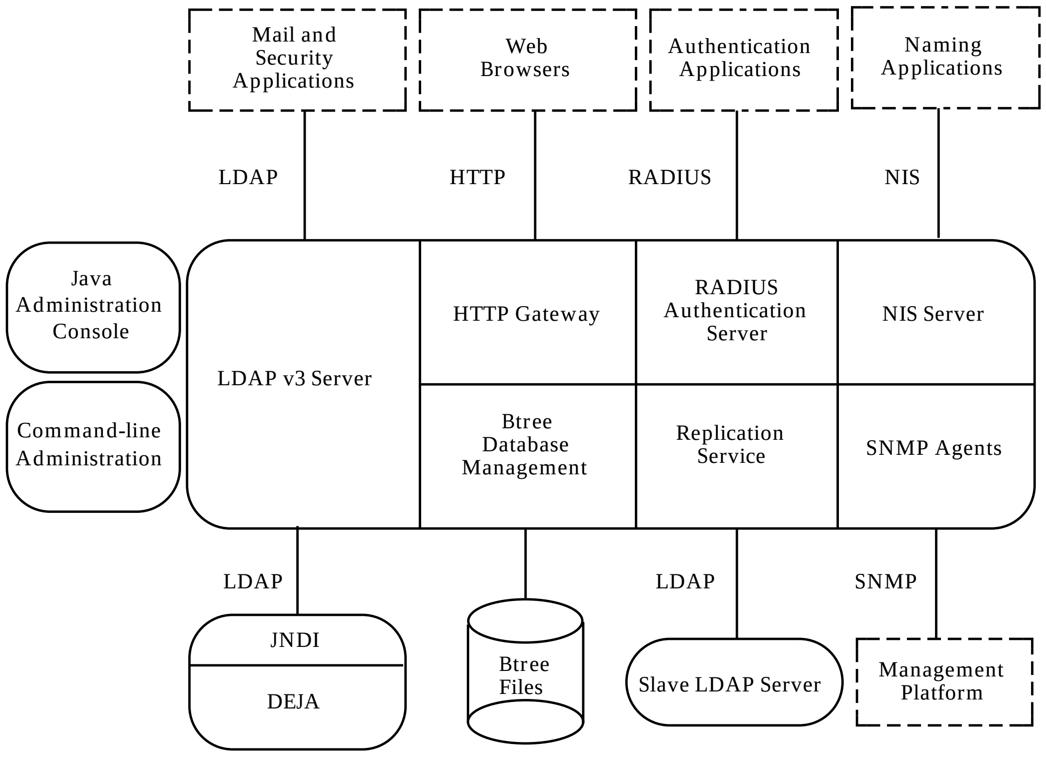


Figure 2–1 Sun Directory Services Architecture

In Figure 2–1, the items shown in dashed boxes are not part of the Sun Directory Services product. They indicate the types of applications that can use information stored in the directory.

Sun Directory Services Components

This section briefly describes the components of the Sun Directory Services. When necessary, cross-references to sections that provide in-depth information on the component are provided.

LDAP Server

The Lightweight Directory Access Protocol (LDAP) is a lighter version of the Directory Access Protocol defined in the X.500 standards, particularly suited to TCP/IP networks.

Sun Directory Services 3.1 implements *RFC 2251 Lightweight Directory Access Protocol (v3)*, the LDAPv3 standard which provides the following enhancements over LDAP v2:

- Enhanced security
- A user-friendly referral mechanism
- Multi-language support for attributes
- Schema retrieval, see “Displaying the Current Schema” on page 126
- The server can be queried on its capabilities, see “Displaying Server Characteristics” on page 55

The LDAP server function is performed by the `dsservd` daemon. The `dsservd` daemon is the component that accesses the database files that hold the directory information, and communicates with directory clients using the LDAP protocol.

LDAP Replication Service

The replication service supplied with the Sun Directory Services enables you to set up a replication strategy for your directory service. You can use replication to share the load of client requests over several servers, and also to reduce overall communication costs by keeping network traffic local.

Replication can apply to a subtree, a particular entry, or a subset of attributes of an entry in the Directory Information Tree.

Sun Directory Services provides two replication daemons, `dspushd` and `dspulld`. The `dspushd` daemon pushes updates from the master server to the slave server, whereas the `dspulld` daemon pulls updates from the master. With the former, the master server manages the replication schedule, whereas with the latter, the slave manages the replication schedule. The communication protocol used by both daemons is LDAP.

Sun Directory Services also includes an NIS replication process associated with the NIS server.

NIS Server

The Network Information System (NIS) is a naming service that offers a way of identifying and locating users and resources on the network.

The NIS server provided with the Sun Directory Services overcomes some of the limitations of a classic NIS naming service, namely:

- A legacy NIS server propagates whole tables, not just updates
- A legacy NIS server cannot handle very large tables, that is tables that hold over 50,000 to 60,000 entries

The NIS server component of the Sun Directory Services can replace an existing NIS server to integrate easily into an existing NIS network. This is done by transferring the information stored in NIS tables on the NIS server into the LDAP directory database. This avoids duplication of user and host information in several databases.

All the information held in NIS tables is mapped onto LDAP object classes and attributes. This mapping can be configured for a particular environment.

The NIS server function of the Sun Directory Services is performed by the `dsservd` daemon. This component responds to NIS requests from user applications. For example, during an `rlogin` operation, it converts the hostname to the IP address of the remote machine.

As in the standard NIS environment, users can change their password using the `passwd` command.

The `dsservd` daemon can act as an NIS master or an NIS slave. As a master, it propagates NIS tables to a slave NIS server. As a slave, it receives propagation requests from a NIS server.

Replication between an LDAP/NIS server and a legacy NIS server is performed using the `dsypxfrd` daemon. For details on how NIS replication is performed by Sun Directory Services, see “Propagating NIS Tables” on page 91.

Chapter 6 explains how to integrate the Sun Directory Services into an existing NIS network, and describes the NIS-to-LDAP information mapping.

RADIUS Server

The RADIUS server component provides user authentication and accounting services. Remote Access Dialup User Service (RADIUS) is the protocol used by Network Access Servers (NAS) to authenticate remote users who connect to the network.The information provided to the NAS in the access request is checked against the information stored in the directory.

All NAS devices available on the market can be configured to use the RADIUS server in Sun Directory Services to authenticate remote users.

The RADIUS server function is performed by the `dsradiusd` daemon. A full description of the RADIUS architecture is provided in Chapter 7.

Administration Console

The Admin Console is a Java tool that enables you to configure, maintain, and monitor the Sun Directory Services. You can use the Admin Console from any machine on your network if there is an HTTP server on the machine where you have installed the Sun Directory Services. If you do not have an HTTP server, you can download the Sun WebServer™ free of charge from `http://www.sun.com/webserver`.

If you do not have a web server colocated with the directory server, you can view the Admin Console locally by pointing a browser at the HTML files, or you can use the application version of the Admin Console.

Refer to "Displaying the Admin Console" on page 34.

Configuration Files

Directory configuration information is stored in a set of configuration files:

- `dsserv.conf` holds the main configuration information
- `dsserv.oc.conf` holds the object class definition part of the schema
- `dsserv.at.conf` holds the attribute definition part of the schema
- `dsserv.acl.conf` holds access control configuration information

Three versions of these files are stored:

- `/etc/opt/SUNWconn/ldap/current` holds the current configuration.
- `/opt/SUNWconn/ldap/default` holds the default configuration that was defined when the software was installed. To revert to a default configuration, choose Restore Config from the Server menu in the Admin Console.
- `/etc/opt/SUNWconn/ldap/previous` holds the previous configuration.

The directory server daemon, `dsservd`, reads the current configuration when it is started. If you change the configuration while `dsservd` is running, you must either click the Refresh button in the Admin Console Status section, or restart the `dsservd` daemon for the configuration to be re-read.

If you make a modification to the configuration, the files in the `current` directory are copied to the `previous` directory. The files are copied only once, and not after every change, so that you have a copy of the configuration that was in effect before you made any modifications.

The Admin Console saves your changes in the configuration files in the `current` directory whenever you click Apply. If you are working in the main window in the browser, when you click Apply, you are prompted to stop and restart the daemon. However, if you are working in a sub window, after you click Apply, you must remember to stop and restart the daemon for your changes to take effect.

The Status section in the Admin Console displays a message indicating when the configuration for a service has changed. This indicates that the service needs to be restarted or refreshed for the changes to be taken into account.

Java Directory Editor (Deja)

Deja is a tool that provides a way to modify and browse the directory content. The search, read, and write permissions granted to a user are determined during the login phase. Deja can run as an applet displayed through the HotJava browser, or locally as an application.

For a complete description of Deja and how to use it, refer to the *Sun Directory Services 3.1 User's Guide*.

Btree Database

The database provided with Sun Directory Services is a proprietary Berkeley Btree database, optimized for directory access. It can hold up to 1 million entries.

A number of command-line utilities are provided for managing the information in the database and reclaiming disk space. Refer to "Data Management" on page 79 for information on these utilities.

Web Gateway

The web gateway provides an end-user interface to an LDAP directory from any web browser. You can use this interface to browse the directory, to search for and read entries, and to modify some directory information.

The web gateway works as an HTTP server. It must be located on the same machine as the LDAP directory server `dsservd`. The web gateway uses configurable templates to display directory information to users.

The web gateway daemon is `dswebd`. For information on starting, stopping, and configuring the web gateway, refer to Chapter 10. For information on using the web gateway, refer to *Sun Directory Services 3.1 User's Guide*.

Administration Utilities

Sun Directory Services is supplied with a complete set of tools for administering the directory service software and the directory content, including monitoring and statistics. The tasks that you can perform with these commands are described in "Data Management" on page 79, and "Directory Maintenance" on page 80.

A generic data loading utility, `dsimport`, is also provided. It converts any text database, including NIS tables, to LDAP objects using a configurable mapping mechanism. For details on `dsimport`, refer to Chapter 5.

Equally, you can back up the directory information base by exporting its contents to a file using the `dsexport` utility.

SNMP Agents

The Sun Directory Services is provided with two SNMP agents:

- An SNMP agent that supports the Mail And Directory MANagement (MADMAN) standard (*RFC 1565 Network Services Monitoring MIB* and *RFC 1567 X.500 Directory Monitoring MIB*), called `dsnmpserv`
- An SNMP agent called `dsnmprad` that supports the RADIUS Accounting Server Management Information Base (MIB) described in *draft-ietf-radius-acc-servmib-01.txt*, and the RADIUS Authentication Server MIB described in *draft-ietf-radius-auth-servmib-01.txt*

For the full list of management statistics you can obtain from the SNMP agents, refer to "Monitoring Directory Services with SNMP" on page 204.

Sun Directory Services Security

The Sun Directory Services provide security protocols in the LDAP server and password encryption mechanisms. However, when the Admin Console is used remotely, the communication between the client machine and the server machine is not encrypted. In this case, authentication is provided by the login procedure alone.

Security Protocols in the LDAP Server

The LDAP server supports the following security protocols:

- Simple Authentication Security Layer (SASL)
- Secure Socket Layer (SSL)

These security features are optional. By default, clients bind to the directory using a simple bind in Insecure mode.

SASL

The SASL protocol is used to provide strong authentication in the bind process through an exchange of tokens. Sun Directory Services supports the CRAM MD5 authentication mechanism. It also supports the EXTERNAL mechanism when the SSL library is installed on the server, and the server is configured to support TLS security.

Secure Socket Layer (SSL)

The SSL protocol is used to provide secure connections between the directory server and directory clients.

The Sun Directory Services implementation of SSL functions in two modes:

- Transport Layer Security (TLS)
- SSL on Specific Port

The SSL on Specific Port mode uses a dedicated port, by default port 636. With the TLS security mode, at any time during an LDAP session you can use the Start TLS extended operation to open a secure connection. When using the Start TLS operation, the client can perform:

- A simple bind
- A SASL protected bind with CRAM-MD5
- A SASL protected bind based on the authentication of the underlying SSL connection.

Both the TLS and SSL on Specific Port modes require an SSL key to authenticate the server. This key is specified using the IP address of the host machine. In both modes it is also possible to configure the server to authenticate clients.

SSL security is available only if the SSL and SKI (Sun Certificate Manager) libraries are available on the server where Sun Directory Services is installed. For details on prerequisites, refer to the installation instructions.

Note - Due to legal restrictions in certain countries, SSL is not available worldwide.

RADIUS Server Encryption

The RADIUS server provided with Sun Directory Services is fully compliant with *RFC 2138 Remote Authentication Dial In User Service* that defines the RADIUS protocol. In the RADIUS protocol, passwords passed between the Network Access Server (NAS) and the RADIUS server are encrypted. The encryption mechanism is MD5 XORing with a shared secret.

Password Encryption

Directory entries can contain user password attributes that are used to authenticate the user to the directory. By default, the values of such attributes are stored in a protected format, identified by the keyword {sunds} in the server configuration file. The encryption algorithm permits the use of the CRAM-MD5 authentication mechanism.

You can also encrypt user passwords using the `crypt(3)` encryption algorithm, which is the algorithm commonly applied to passwords stored in the `/etc/passwd` file. This algorithm is incompatible with the CRAM-MD5 authentication mechanism. This encryption method is identified by the keyword {crypt} in the server configuration file.

See “Configuring Security” on page 39 for details of how to specify whether or not passwords are stored in an encrypted format.

CHAPTER 3

Planning a Directory Service

This chapter explains the global decisions you need to make about your directory service before you can start configuring individual components.

These are the things that you need to consider:

- Type of information to store, for example, entries describing mail users, remote users connecting through a NAS, NIS tables, equipment connected to the network
- Overall volume of information, and how to divide it up into naming contexts
- How many servers you require
- The information and services expected from each server
- The overlap of information between servers
- Which servers will hold master copies of replicated information
- Replication methods

Types of Information

You must define the kind of information that you want to store in the directory. The schema must include the object classes and attributes required to store your information. If the default schema does not cover all your needs, refer to “Modifying the Schema” on page 127 for details on how to create the object classes and attributes that you need.

The default schema will allow you to store information on resources such as:

- Users
- Remote RADIUS users
- Devices (hosts, printers, Network Access Servers)

- NIS naming service
- Java objects
- Document references

For a complete list of the type of objects you can create in the directory with the default schema, refer to “Object Class Reference” on page 131.

When you are looking for an object class that matches the characteristics of a resource you want to describe in a directory entry, you must make sure that:

- The object class offers or inherits all the attributes you need to describe your resource
- You can assign a value to all the mandatory attributes offered or inherited by that object class

For example, XYZ Corporation has a proprietary on-line directory that provides the following information about every employee:

- Name (first name, last name)
- E-mail address
- Telephone number
- Job function

In the default schema, the object class that allows you to store all these attributes for a person is inetOrgPerson. This object class contains the optional mail attribute for storing an e-mail address. It inherits from the organizationalPerson and Person object classes. These object classes provide the necessary attributes for storing employees’ names, telephone numbers and job functions.

Creating Naming Contexts

Having defined the type of information you want to store in the directory, you must decide how you are going to divide it into naming contexts. You can choose to divide the information according to:

- Geographical boundaries, for example per company site
- Corporate organization
- DNS mail domains
- User types
- Service types (NIS, RADIUS, for example)
- A combination of the above: for example, when you enable the NIS service, by default one naming context is created for people, and another for services. If you enable the RADIUS service, it is recommended to have one naming context for

remote users, and another for network access servers (NAS). You can also create a naming context for each corporate location.

The logic you follow to divide your information into naming contexts can be defined regardless of the location of the information because you can replicate any part of any naming context to any server in your network.

However, you should make sure that any given resource has just one master entry in the directory. Otherwise, you will lose the main advantage of having one unique information repository.

Example: Naming Contexts in XYZ Corporation

The XYZ Corporation is an international pharmaceutical company with headquarters in Boston, USA. They have two manufacturing operations, one in Boston and one in London. There are two research groups, in New York and Paris colocated with marketing and sales. The sales organization also has an office in Hong Kong. The Human Resources function is represented in Boston for the US sites and in London for the European sites and the rest of the world (RoW).

Figure 3–1 shows the functional structure of XYZ Corporation.

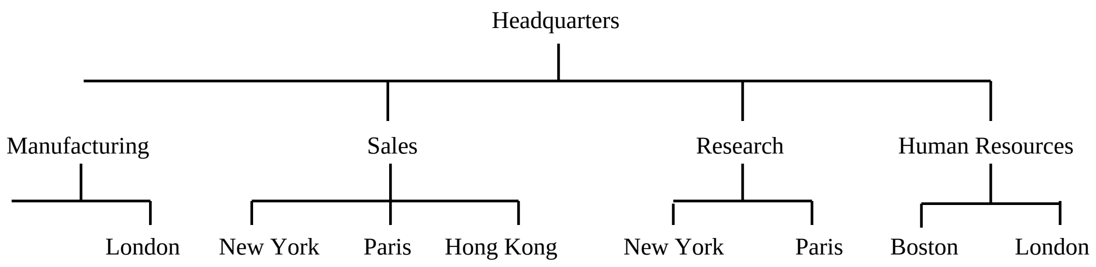


Figure 3–1 Functional Structure of XYZ Corporation

Figure 3–2 shows the geographical structure of XYZ Corporation.

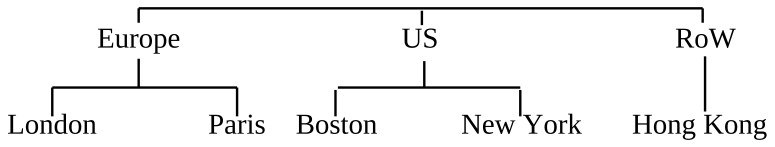


Figure 3–2 Geographical Structure of XYZ Corporation

Because the main functions within the XYZ Corporation are located on different sites, neither an organizational DIT structure nor a functional DIT structure completely meets the company’s directory structure needs. Also, the XYZ Corporation uses the NIS naming service to centralize information on mail users and network resources. Because this information is critical to the company’s operation, the network management team wants to maintain it centrally, regardless of organizational or geographical boundaries. However, they are willing to enforce a per site maintainance of information considered to be less critical, and likely to be updated often. This is the case for information on users who are granted remote access to the network through a NAS.

The result is the DIT structure shown in Figure 3–3.

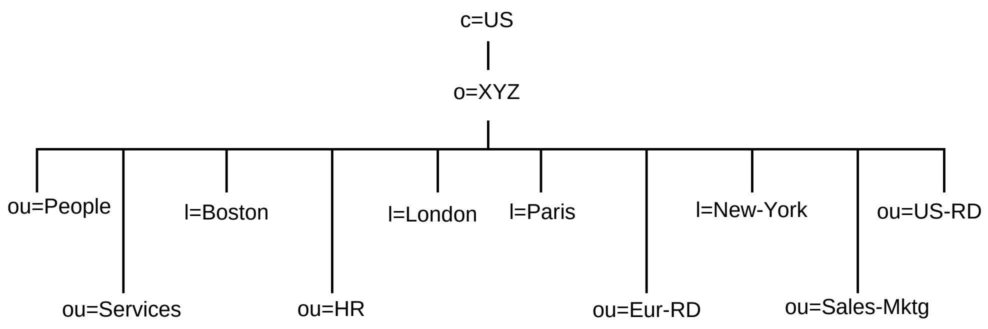


Figure 3–3 DIT Structure for XYZ Corporation

In this DIT structure, the corporation is divided into six organizational units (ou) and four localities (l) corresponding to ten naming contexts. Table 3–1 shows the RDN of each naming context and the servers on which it is stored. Table 3–2 shows which is the master server for naming contexts that are replicated to other servers.

Location of Information on Servers

You must determine on which servers in your network to store the naming contexts you create. These are the constraints that you will need to take into account:

- Number of servers
- Location of servers
- User requirements: make sure users get the information they need most often from the closest server to reduce communication costs
- Size of the directory database: the maximum number of entries in each data store is 1 million (though a server can hold several data stores)

Based on the constraints imposed by your network, and the needs of users, you can work out where you want to store information. This decision can be made regardless of overlaps of information between servers. These overlaps will be handled by setting up a replication strategy.

For example, the XYZ Corporation has a directory server on each site except in Hong Kong. Table 3–1 shows the naming contexts held by each server.

TABLE 3–1 Naming Contexts Location in XYZ Corporation

Server	Naming Contexts
boston	ou=XYZ, c=US
	ou=People, ou=XYZ, c=US
	ou=Services, ou=XYZ, c=US
	ou=HR, ou=XYZ, c=US
newyork	l=New-York, ou=XYZ, c=US
	ou=US-RD, ou=XYZ, c=US
	ou=Eur-RD, ou=XYZ, c=US
	ou=Sales-Mktg, ou=XYZ, c=US
london	l=London, ou=XYZ, c=US
	ou=People, ou=XYZ, c=US
	ou=Services, ou=XYZ, c=US
	ou=HR, ou=XYZ, c=US
paris	l=Paris, ou=XYZ, c=US
	ou=Eur-RD, ou=XYZ, c=US
	ou=US-RD, ou=XYZ, c=US
	ou=Sales-Mktg, ou=XYZ, c=US

Setting Up a Replication Strategy

To set up a replication strategy, you need to identify the number of locations where the same information is present, determine which location will hold the master copy, and define the most efficient replication process between servers.

Simple and Cascading Replication

If you have two locations that hold the same information, you will have a simple replication with one master and one slave. If you have three or more locations that hold the same information, you can have a simple replication with one master and two or more slaves, or a cascading replication.

In a cascading replication, there are several levels of information replication: a replica copy on a slave server is in turn replicated to another slave. For information that is replicated in many locations, a cascading replication has the advantage of reducing the network traffic originating from the master server by sharing the load with the first level of slave servers.

Replication Methods

Sun Directory Services replication is based on the LDAP protocol. However, when the NIS service is enabled on a Sun Directory Services server, NIS replication can take place between this server and legacy NIS servers.

The NIS replication feature of Sun Directory Services offers a smooth transition from a classic NIS naming service to an integrated naming and directory service.

LDAP replication has the following advantages over NIS replication:

- Partial replication is possible: you can choose the part of the DIT that you want to replicate, and you can also replicate just the changed attributes of modified entries
- Better scalability: NIS replication is likely to fail if the number of entries replicated is in excess of 50,000.

Therefore, when the Sun Directory Services product is installed on several servers used as NIS servers, NIS replication should be disabled in favor of LDAP replication. For details, refer to "Configuring the NIS Service" on page 90.

There are two replication modes for propagating changes between master servers and slave servers:

- Master-based replication: the master server owns the synchronization schedule and pushes modifications to its slaves
- Client-based replication: the slave server owns the synchronization schedule, and pulls modifications from the master server

---

Caution - If you have both LDAP v2 and LDAP v3 servers in your environment, you may lose LDAP v3 specific information when replicating from an LDAP v3 server to an LDAP v2 server. For example, if replicated information makes use of language tags in attributes, all replication targets must support LDAP v3. Refer to the product release notes for details of migration from Sun Directory Services 1.0 to Sun Directory Services 3.1.

Example: Replication in the XYZ Corporation

Table 3–1 shows that some of the naming contexts in the XYZ Corporation are held on several servers in order to provide faster access and to reduce the connection costs by keeping information local.

One server holds the master copy, the others are replicas.

Table 3–2 shows the replication strategy for each naming context in the XYZ Corporation directory service.

TABLE 3–2 Replication Strategy for the XYZ Corporation

Naming Context	Master Server	Slave Server
ou=People, ou=XYZ, c=US	boston	london
ou=Services, ou=XYZ, c=US	boston	london
ou=HR, ou=XYZ, c=US	boston	london
l=Boston, ou=XYZ, c=US	boston	
l=New-York, ou=XYZ, c=US	newyork	
ou=US-RD, ou=XYZ, c=US	newyork	paris
ou=Sales-Mktg, ou=XYZ, c=US	newyork	paris
l=London, ou=XYZ, c=US	london	
l=Paris, ou=XYZ, c=US	paris	
ou=Eur-RD, ou=XYZ, c=US	paris	newyork

The replication method used to replicate changes between master servers and slave servers is LDAP.

Referrals

A referral system ensures that if an entry cannot be found locally, the directory server can pass a referral back to the client with a request to contact another directory server. Most LDAP client libraries automatically follow referrals and resubmit the user’s original request.

There are two ways of defining referrals in Sun Directory Services:

- Default referral for the directory server
- Referral entries

The default referral is a parameter that you configure for each directory server. It is a pointer to another *directory server*. Usually, the referral server holds a higher node in the DIT than the one held by the current server, in order to widen the scope of the search.

A referral entry can be created at any level in the DIT to point to any subtree of the DIT held on a different *data store*. The value of a referral entry is a URL.

Note - Do not create referrals from LDAP v3 servers to LDAP v2 servers. Most client libraries cannot follow a referral to an LDAP v2 server when the initial request was submitted to an LDAP v3 server.

Referral entries can be used to overcome the one million entry limit for a data store. It is possible to create a data store that holds referral entries directly under the root entry. These entries point to subtrees of the global DIT held on different data stores on the same server, or on different servers.

This principle is illustrated in Figure 3–4.

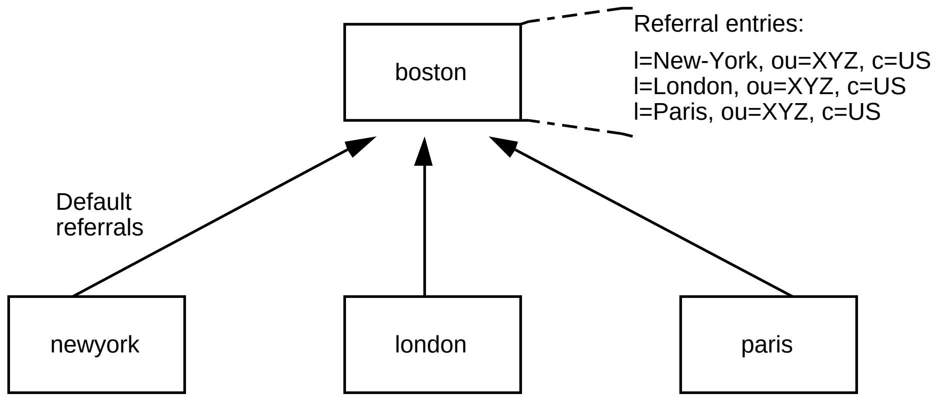


Figure 3–4 Referrals in a Multi-Million Entry Directory

The newyork, london and paris servers have a default referral to the o=XYZ, c=US data store held on the boston server. The boston server contains the following three referral entries:

- l=New-York, ou=XYZ, c=US
- l=London, ou=XYZ, c=US
- l=Paris, ou=XYZ, c=US

A referral entry contains the following attributes:

- objectClass = referral

- ref = "ldap://*hostname*[:*port*]/*DN*"
- The naming attribute of the referenced data store, also used to name the referral entry, to avoid using the ref attribute as the naming attribute

For example, the l=New-York, ou=XYZ, c=US referral entry contains:

```
DN: l=New-York,ou=xyz,c=US
objectClass: referral
ref: ldap://newyork/ou=New-York,ou=XYZ,c=US
l: New-York
```

If a search operation in the l=New-York, ou=XYZ, c=US tree fails, the LDAP protocol follows the default referral and searches the ou=XYZ, c=US tree. If a search is started on the ou=XYZ, c=US tree, the LDAP protocol follows the URLs provided in the referral entries to search all referenced data stores.

CHAPTER 4

Configuring a Directory Server

This chapter explains how to configure a directory server using the Admin Console. It describes the following tasks:

- Starting the Admin Console
- Performing a minimal configuration
- Configuring security
- Configuring LDAP parameters
- Creating and configuring a data store
- Indexing directory data
- Creating and modifying access control rules
- Configuring logging

This chapter also describes the default configuration for Sun Directory Services.

However, before you configure each directory server individually, you must define a global plan for your directory service. Chapter 3 guides you through the decisions you need to make to set up an overall plan that covers your directory needs, naming service needs, and authentication and accounting needs.

Use the table below to find the section that describes the tasks that are not covered in this chapter.

Task	**Section**
Configuring the NIS service	"Configuring the NIS Service" on page 90
Configuring the RADIUS service	"Configuring the RADIUS Server" on page 111

Task	Section
Viewing and modifying the schema	Chapter 8
Configuring replication	Chapter 9
Configuring the web gateway	"Configuring the Web Gateway" on page 211

Logging In for the First Time

If you have installed Sun Directory Services for the first time, you must specify a password for the administrator user. To do this, type the following command as `root`:

```
# /opt/SUNWconn/ldap/sbin/dsadmintool
```

You are prompted to provide a password. When you have done this, the Admin Console is displayed. You are logged in as user `admin`, with the password that you specified. You can change this name and password at any time from the Security section of the Admin Console. This procedure is explained in "Configuring Security" on page 39.

All of the examples in this manual use the administrator name *admin* and the password *secret*.

Note - When you log in to the Admin Console for the first time, you cannot start `dsadmintool` as a background process.

Displaying the Admin Console

Before you follow the instructions in this section to display the Admin Console, make sure that the Sun Directory Services administration daemon, `dsadm`, is running. If it is not, start it as explained in "Starting the Sun Directory Services" on page 203.

This section explains three methods of displaying the Admin Console:

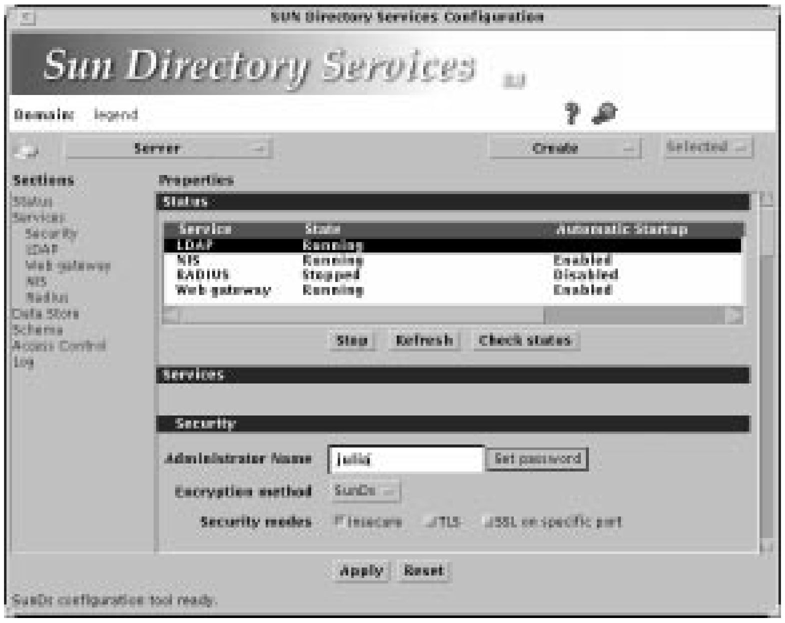
- Starting the application version locally
- Through a web server. You can download the Sun WebServer free of charge from `http://www.sun.com/webserver`
- Without a web server, by pointing a web browser at the Sun Directory Services administration files

▼ To Start the Admin Console Application

If you do not have a web server colocated with Sun Directory Services, you can also run the Admin Console locally as an application. To start it, type the following command as `root`:

```
/opt/SUNWconn/ldap/sbin/dsadmintool &
```

The Admin Console is displayed, as shown in Figure 4–1.



Domain: legend

Service	State	Automatic Startup
LDAP	Running	
NIS	Running	Enabled
RADIUS	Stopped	Disabled
Web gateway	Running	Enabled

Administrator Name: julia
Encryption method: SunDs
Security modes: Insecure TLS SSL on specific port

SunDs configuration tool ready.

Figure 4–1 Admin Console Main Window

To Display the Admin Console Remotely through a Web Server

1. **Create a symbolic link from the documentation root directory for your web server to** `/opt/SUNWconn/html`**:**

```
prompt% cd docroot_dir
prompt% ln -s /opt/SUNWconn/html sds
```

where *docroot_dir* is the documentation root directory for your web server.

Your web server must be running on the same machine as the `dsservd` daemon.

2. **If you are using the HotJava browser, change the default security level by editing the `.hotjava/properties` file in your home directory.**

   Use any text editor to change the following line from:

   ```
   hotjava.default.security=medium
   ```

   to:

   ```
   hotjava.default.security=low
   ```

   If you have not used the HotJava browser before, you must create the `.hotjava/properties` file and add the line `hotjava.default.security=low`.

3. **Start the HotJava browser on any machine, and point it at the Admin Console:**

   ```
   prompt% /usr/dt/bin/hotjava http://hostname/sds/dsadmin.html
   ```

   where *hostname* is the hostname of the machine running the Sun Directory Services and the web server.

4. **Log in to the Admin Console using the name and password of the administrator user.**

   The Admin Console is displayed.

To Display the Admin Console Locally without a Web Server

1. **If you are using the HotJava browser , change the default security level by editing the `.hotjava/properties` file in your home directory. Use any text editor to change the following line from:**

```
hotjava.default.security=medium
```

to:

```
hotjava.default.security=low
```

If you have not used the HotJava browser before, you must create the `.hotjava/properties` file and add the line `hotjava.default.security=low`.

2. **Start the HotJava browser with the following path to point it at the administration files:**

```
prompt% /usr/dt/bin/hotjava file:/opt/SUNWconn/html/dsadmin.html
```

3. **Log in to the Admin Console using the name and password of the administrator user.**

   The Admin Console is displayed.

Minimal Configuration

When you install the Sun Directory Services, most configurable characteristics are given default settings that enable you to start and run a directory server. However, you *must*:

- Give a name to the naming context held in the data store. For details, see “Configuring the Data Store” on page 41.
- Specify the location of the data store on the disk. For details, see “Configuring the Data Store” on page 41.

When you have specified this information your directory server has a default configuration.

Default LDAP Configuration

The LDAP directory service has the following characteristics:

- The port used for LDAP communications is 389.

- Alias dereferencing on bind operations is enabled.
- Searches are limited to 5000 entries or 3600 seconds (1 hour). A search stops when the first of these limits is reached.
- The data store is in `/var/opt/SUNWconn/ldap/dbm`.
- Log files are stored in `/var/opt/SUNWconn/ldap/log`.
- 5000 entries are cached. A cache file of 100,000 bytes is created for each data store.
- Default indexing is used, as described in "Indexing Directory Data" on page 46.
- Passwords are stored in encrypted format.
- Default access control is used, as described in Code Example 1–1.
- The schema is checked for each directory operation, as described in "Schema Checking" on page 128.
- Security on LDAP connections is disabled.
- The directory contains no entries (the name and password for the administrator are stored in the configuration file).
- There are no referrals to other servers.

Default RADIUS Configuration

The RADIUS service has the following characteristics:

- The ports used for RADIUS communications are port 1645 for authentication traffic, and port 1646 for accounting traffic. However, recent standardization work suggests to use port 1812 for authentication traffic, and port 1813 for accounting traffic. If you want to use these ports, you can specify them using the Admin Console.
- The RADIUS server is disabled.

Default Web Gateway Configuration

The web gateway has the following characteristics:

- The port used by the web gateway is 1760.
- The templates used to configure the display of directory information are `/etc/opt/SUNWconn/ldap/current/locale/C/dswebtmpl.conf`.

Default NIS Configuration

The NIS service is disabled.

Starting and Stopping Services

At any time, you can independently start or stop the following components of Sun Directory Services from the Admin Console.

- LDAP
- NIS
- RADIUS
- Web gateway

The Admin Console also enables you to specify whether you want any of these components except LDAP to restart automatically when the server daemon, `dsservd`, is restarted.

1. **From the Admin Console main window, in the Status section, select the service that you want to start or stop.**

2. **For each service, verify the status for Automatic Restart at boot time. Change it as necessary.**

3. **Click Apply to save your changes.**

Configuring Security

You can configure the security level you require for your directory server by setting the administrator's name and password, defining the encryption method for all passwords stored in the directory, and specifying the security mode for your server.

Specifying the Administrator Name and Password

You can change the name and password of the administrator user at any time.

1. **From the Admin Console main window, in the Security section, specify the name and password for the directory administrator.**

   The administrator name and password are stored in the configuration file `dsserv.conf`, so that the administrator always has access to the directory. This is necessary so that the administrator can solve problems with access control, for example.

By default, the administrator's password is encrypted using the default encryption method. If you want the password to be stored unencrypted, choose None from the Encryption method menu.

2. **Click Apply to save your changes.**

Specifying the Encryption Method

The encryption method you specify is used to encrypt the userPassword attribute that you can store in directory entries. There are two encryption methods:

- sunds: a reversible encryption mechanism compatible with CRAM MD5 authentication
- `crypt(3)`: the encryption mechanism commonly applied to passwords stored in the `/etc/passwd` file. This method is not reversible and is incompatible with the CRAM MD5 authentication mechanism.

1. **From the Admin Console main window, in the Security section, select the Encryption method for passwords from the menu button.**

   The possible choices are: None, sunds, crypt.

2. **Click Apply to save your changes.**

Specifying the Security Mode

You can specify a security mode that guarantees that authentication takes place between the server and the client during the bind process, or that the entire communication between the server and the client is encrypted. You can also allow insecure binding. These options are not mutually exclusive. The security mode used for a particular connection is actually negotiated between the client and the server.

1. **From the Admin Console main window, in the Security section, specify the Security modes supported by the server.**

   The options are:

   - Insecure
   - TLS: establishes a secure connection at any time during an LDAP session by calling the Start TLS extended operation
   - SSL on Specific Port: opens a secure connection on the specified port (by default, port 636)

   You can enable several options.

2. **Specify the following parameters:**

   If you have selected TLS or SSL on Specific Port:

- The SSL key package (usually the IP address of the host)
- Whether you want the server to authenticate clients: set the SSL Client Verification flag to On

If you have selected SSL on Specific Port:

- A port number to use for SSL connections

3. **Click Apply to save your changes.**

Note - Due to legal restrictions in certain countries, SSL is not available worldwide.

Configuring LDAP Parameters

In the Admin Console, use the LDAP section under Services to configure the following properties of the LDAP server:

- The port used by the directory server daemon, `dsservd`.
- Whether alias entries are to be dereferenced when a bind request is received.
- Search limits (number of entries and time).
- The default directory server for referrals, and the LDAP port number of the referral server. Give the name of a server that holds knowledge information on subtrees not managed by the current server. This should be a server that holds a broader view of the same tree, to allow searches in the current tree to be extended through the referral. To define referrals to different trees, use referral entries.

Configuring the Data Store

To configure a data store, you *must* specify the distinguished name of the naming context that identifies the data store, and the name of the directory where the database files reside. Optional configuration information includes:

- Additional naming contexts stored (see “Naming Contexts” on page 43)
- Cache size (see “Caching” on page 43 for information about caching)
- Congestion thresholds (see “Congestion Thresholds” on page 44 for information about the congestion monitoring system and setting thresholds)

- Which attributes are indexed (see “Indexing Directory Data” on page 46 for information about indexing)
- Whether any of the naming contexts held in the data store are replicated to other servers (see “Setting Up Replication” on page 196)

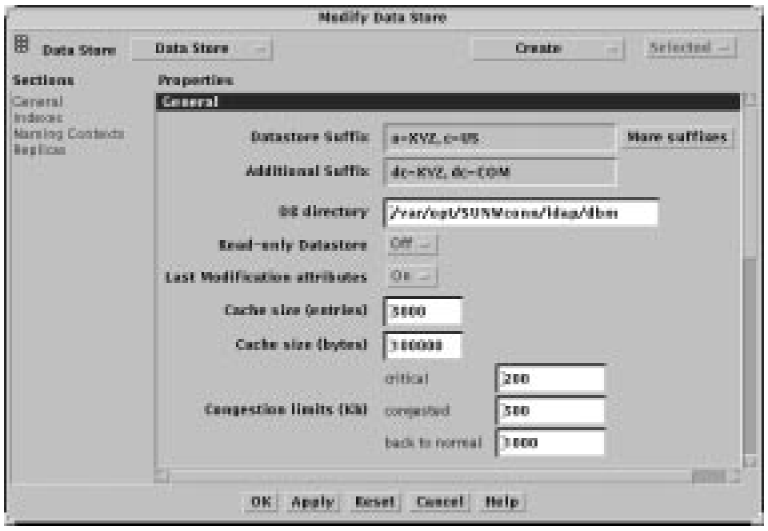
“To Create a Data Store” on page 44 gives step-by-step instructions for creating a new data store. “To Modify a Data Store or Naming Context” on page 46 explains how to modify an existing data store.

▼ To Rename the Default Data Store

In your initial configuration, you must specify the distinguished name of the naming context that identifies the data store on the server, and the directory where the data store files are held.

1. **In the Admin Console main window, highlight the default data store o=XYZ, c=US, in the Data Store section and choose Modify from the Selected menu.**

   Alternatively, you can double-click on the default data store o=XYZ, c=US. The Modify Data Store window is displayed, as shown in Figure 4–2.



Datastore Suffix: o=XYZ, c=US
Additional Suffix: dc=XYZ, dc=COM
DB directory: /var/opt/SUNWconn/ldap/dbm
Read-only Datastore: Off
Last Modification attributes: On
Cache size (entries): 1000
Cache size (bytes): 100000
Congestion limits (Kb): critical 200, congested 500, back to normal 1000

Figure 4–2 Modify Data Store Window

2. **Modify the data store suffix to reflect the distinguished name of the naming context that the directory server will store.**

   If you intend to initialize the NIS service, you must modify the suffix made up of dc elements so that this naming context can contain the NIS domain managed by the server.

If you want the data store to hold more than one naming context, you can specify additional naming suffixes by clicking the More Suffixes button. If necessary, use the Set button to specify the distinguished name. "Using the DN Editor" on page 55 explains how to enter a DN in the Admin Console.

3. **Specify the path name to the directory where the data store is to be held.**

   For a default configuration, you do not need to specify any other information. For more information about creating a data store, see "Creating or Modifying a Data Store" on page 43.

Creating or Modifying a Data Store

This section gives information on the options you can configure when creating or modifying a data store. It also guides you through the steps for performing these tasks.

Naming Contexts

A data store can be identified by up to four suffixes. A suffix corresponds to a naming context. The term naming context also applies to any subtree under a data store suffix. In the Admin Console, the Data Store section displays the naming contexts held in the data store. This list contains the data store suffixes plus any naming contexts created under the data store suffixes.

The distinguished name of a naming context corresponds to the base DN you specify when you perform a directory operation such as a search, or when you configure an LDAP client application.

You can specify a DN that conforms to the OSI tree structure (for example o=XYZ, c=US), or a DN that reflects your organization's DNS naming structure, through a sequence of domain components (for example dc=XYZ, dc=com).

A data store can contain a mixture of master and replica (slave) naming contexts, some or all of which can be replicated to other servers. For information on replication, refer to Chapter 9.

Note - You can modify the DN of a data store only when the data store is empty.

Caching

As information is retrieved from the directory it is saved in the cache. When the cache is full, the oldest entry is discarded to make room for new information. Retrieving information from the cache is faster than retrieving entries from the database, but a large cache occupies more memory.

Default value for cache size is 5000 entries.

Congestion Thresholds

Congestion thresholds ensure that the directory does not become overloaded, by preventing new operations from starting when there are insufficient resources.

- When the available disk space reaches the *congested* limit, add operations are no longer permitted, though modify, modify RDN (modrdn), search, read, and delete operations are allowed. The default limit is 500 Kbytes.
- When the database reaches the *critical* limit, only search, read, and delete operations are allowed, and add, modify, and modrdn operations are not permitted. The default limit is 200 Kbytes.
- The restrictions remain in force until the remaining disk space becomes greater than the *back-to-normal* limit. The default limit is 1000 Kbytes.

If your database reaches the congested limit, after deleting unnecessary entries, you must still regenerate the database to optimize the use of disk space. Refer to "Regenerating the Database" on page 81.

Threshold values are given in Kbytes free on the disk holding the data store. The default location for the data store is `/var/opt/SUNWconn/ldap/dbm`.

To Create a Data Store

1. **In the Admin Console, choose Data Store from the Create menu.**

   The Create Data Store window is displayed. It is similar to the Modify Data Store window shown in Figure 4–2.

2. **In the General Section, in the Data Store Suffix field, specify the distinguished name of the naming context stored in this data store.**

   To specify more than one naming context, click the More Suffixes button.

3. **Specify the path name to the directory where this database is to be stored.**

4. **Specify whether the data store is read-only or not.**

   By default, it is not.

5. **Specify whether the last modification attributes should be recorded.**

   When this option is enabled, the creator's DN, a creation timestamp, the modifier's DN, and a modification timestamp are added in the entry. This enables you to search the directory for entries created or modified since a particular date or time. This can be useful if you are synchronizing changes across data stores, or you are using partial replication.

6. **Specify the cache size limit, in entries and in bytes.**

   See “Caching” on page 43 for details.

7. **Specify congestion thresholds.**

   See “Congestion Thresholds” on page 44 for details.

8. **Optionally, you can specify which attributes to index in the database.**

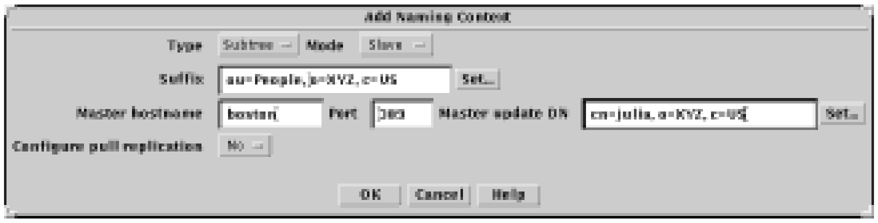
   See “To Create Indexes” on page 47 for details.

9. **In the Naming Contexts section, specify the master and slave subtrees held in this data store.**

   These are naming contexts that are subtrees or objects under the naming context used to name the data store. If you do not specify any naming contexts, the distinguished name of the data store itself is added to the list of master naming contexts automatically, but it is not displayed until you save the naming contexts information. See “Naming Contexts” on page 43 for details.

   a. **Choose Naming Context from the Create menu.**

      The Add Naming Context window is displayed. Figure 4–3 shows the appearance of the window when Slave mode is selected.



Add Naming Context

Type: Subtree   Mode: Slave

Suffix: ou=People, o=XYZ, c=US

Master hostname: boston   Port: 389   Master update DN: cn=julia, o=KYZ, c=US

Configure pull replication: No

Figure 4–3 Add Naming Context window (Slave Mode)

   b. **Specify:**

      - The subtree type (subtree or object)
      - The DN of the subtree or object in the Suffix field
      - The mode (master or slave)

   c. **If the naming context is a replica (slave), specify:**

      - The name of the server from which it is replicated in the Master hostname field
      - The Master Update DN that `dspushd` will use when the master initiates the bind to replicate changes.

- Whether you want the slave to pull updates from the master. If so, see “To Create Replicas” on page 197 for details on how to specify the bind parameters.

d. **Click OK to save the naming context information.**

The new naming context is displayed in the Naming Contexts section.

10. **Optionally, you can replicate any of the stored naming contexts to a different server.**

    If you do, you must create the naming context on the target server following the procedure described in Step 9 on page 45, because the replication procedure does not create it automatically. See “Setting Up Replication” on page 196 for details.

11. **Click OK in the Create Data Store window to save the data store definition.**

12. **Add a directory entry for the root of the data store.**

    See “Creating the Root Entry” on page 61 for details of how to add an entry to the directory. You cannot add any entries to this data store until this root entry exists.

To Modify a Data Store or Naming Context

♦ **To modify a data store, select the data store you want to modify from the Data Store list, and choose Modify Data Store or Modify Naming Context from the Selected menu.**

Alternatively, you can double-click on your selection. The current characteristics of the data store or naming context are displayed. You can modify any part of the configuration, apart from a data store suffix, when the data store is not empty.

Indexing Directory Data

A data store can contain a number of attribute indexes to help optimize the speed of access to directory information. An attribute index is a list of entries containing a given attribute or attribute value. You can index attributes using any of the following matching rules:

- Equality – Optimizes direct access to entries where an exact attribute value is supplied.
- Presence – Optimizes searches with filters specifying the presence of an attribute but no specific value (cn=*, for example).

- Substring – Optimizes searches with filters containing a partially-specified attribute value (cn=adr*, for example). Substring indexing uses all possible combinations of three successive letters. For example, if cn=adrianne, the substring indexes are: adr, dri, ria, ian, ann, nne. This type of index is longer to generate and is the most costly in terms of disk space.
- Approximate – Optimizes searches with approximate match filters. The method used in approximate indexing is to discard vowels.

In a data store having the default characteristics, the following attributes are indexed:

- commonName, surname, mail, mailHost, and givenName are indexed by presence, equality, approximate match and substring match
- uid is indexed by presence and equality
- preferredRfc822Recipient, rfc822Mailbox, cCMailAddress, pROFSAddresses, mSMailAddresses are indexed by presence and equality

The advantage of indexing is that it optimizes access for indexed attributes. The disadvantages are that it uses more disk space, and that adding and modifying entries takes longer.

When you add or modify an entry after an index has been created, the index is automatically updated. However, if you create a new index and the data store already contains entries, those entries are not automatically included in the index. Indexes are not automatically updated when entries are removed from the directory, so the size of the index files does not reduce as entries are removed.

To update all the indexes defined in a data store, choose Refresh Index from the Data Store menu of the data store window (see “Directory Maintenance” on page 80.) Regenerating the indexes for a data store can take several minutes, depending on the number and complexity of the indexes defined. For example, regenerating the default indexes for a data store of 20,000 entries takes approximately five minutes.

To Create Indexes

1. **Whether you are creating a new data store or modifying an existing one, select Index from the Create menu.**

   The Add Index window is displayed.

2. **Specify the name of the indexed attribute and the index types.**

   Use the Set button to select attributes from a list. You can specify several attributes separated by commas. These indexed attributes will have the same index types.

3. **Click Apply to add the index.**

Backing Up and Restoring a Data Store

The backup function enables you to create a backup of the information held in your directory database in text format. The restore option enables you to restore your database from a set of backup files.

To Back Up a Data Store

1. **To back up a data store, double-click on the data store in the Data Store list to display it.**

   The Modify Data Store window is displayed.

2. **Click the Data Store pull-down menu, and select Backup.**

   A selector window is displayed.

3. **Use the selector window to select or create a directory for your backup.**

4. **Click Save.**

To Restore a Data Store

1. **To restore a data store, double-click on the data store in the Data Store list to display it.**

   The Modify Data Store window is displayed.

2. **Click the Data Store pull-down menu, and select Restore.**

   A selector window is displayed.

3. **Select the directory containing the backed up data store that you want to restore.**

   The default selection is the previous backup directory.

4. **Click Load.**

   The directory service is restarted with the restored data store.

Configuring Access Control

Access controls determine who has access to a given directory entry, and what level of access is granted. "Access Control" on page 7 explains how to design an access control policy for your directory. The following sections explain how to configure default access, and add, modify, and delete access control rules.

An access control rule defines the level of access to specific directory information given to a particular user. There are two stages to defining a new access control rule:

- Specify the directory information to which the rule applies. This is the information that you want to protect.
- Specify the level of access granted to each user for this information. From the most restrictive to the most permissive, levels of access are: none, compare, search, read, write.

Access control rules are ordered, with the most specific rules first, followed by more general rules. The first rule in the list that matches the requested operation is applied, the following rules in the list are ignored.

Default access is the level of access granted to entries and attributes for which access control is not specifically defined.

▼ Configuring Default Access

1. **In the Admin Console main window, go to the Access Control section.**
2. **Use the Default Access menu button to select a level of access.**
   By default, this option is set to read.

▼ To Add an Access Control Rule

1. **Choose Access Control from the Create menu.**
   The Create Access Control Rule window is displayed.
2. **Specify the information to which the new rule will apply, as follows:**
   a. **From the Selected Entries menu, select the method of specifying the entries.**
      You can specify entries using a DN-based regular expression, an LDAP filter, or you can specify that the rule applies to all entries.
      - If you selected DN-based regular expression, type the regular expression in the Distinguished name field, or click Set to use the Distinguished Name Editor to specify the regular expression.

If you want to protect only certain attributes within the set of entries defined by the regular expression, click the Attribute Set button and select the attributes to be protected. If you do not specify any attributes, all attributes in the specified entries are protected.

- If you selected LDAP filter, click the LDAP filter Set button to launch the LDAP Filter Editor. Specify the filter, and click Apply.

  If you want to protect only certain attributes within the set of entries defined by the regular expression, click the Attribute Set button and select the attributes to be protected. If you do not specify any attributes, all attributes in the specified entries are protected.

b. **Type the name of an attribute to be protected in the Attributes field.**

   To see a list of attributes, click the Set button. You can specify any number of attributes.

3. **Choose Access Rule from the Create menu.**

   The Add User Rule window is displayed.

4. **Select the Rule type. This defines the set of users to which the rule applies.**

   You can specify a rule for Everyone, DN-based Regular Expression, Self (that is, the entity described by the entry), Address, Domain, or Member Attribute.

   - If you selected Everyone, the rule will apply to all users whose directory entries contain this attribute.
   - If you selected DN-based Regular Expression, specify the regular expression for the set of users to which the rule applies. The rule will apply to all users who bind with a distinguished name that matches the regular expression.

     You can type the distinguished name directly in the field, or you can click Set to use the Distinguished Name editor to construct the distinguished name. See "Using the DN Editor" on page 55 for more information about how to specify a distinguished name.
   - If you selected Address, specify an IP address.

     The IP address can contain wildcards. The rule will apply to all users who bind from the specified IP address.
   - If you selected Domain, specify a domain name.

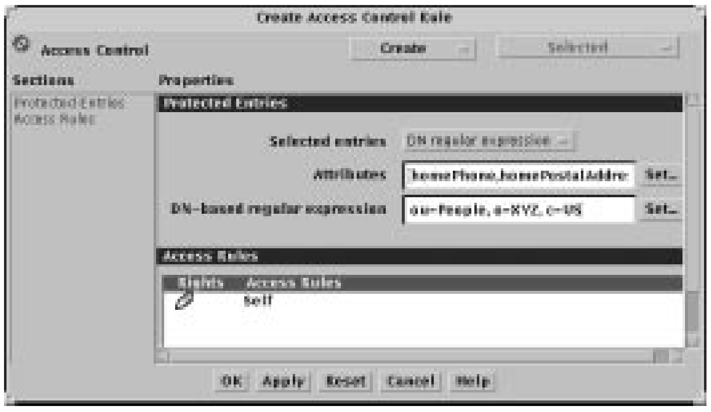
     The domain name can contain wildcards. The rule will apply to all users who bind from the specified domain.
   - If you selected Member Attribute, specify an attribute.

     The rule will allow the DN used in the bind to be added to or removed from the list of members specified by the attribute.

5. **Specify the access rights to be granted to the specified set of users.**

6. **Click Apply to add the rule.**

   You can then define other rules for entries you have selected. When you have created and added all the rules for these entries, click Cancel to dismiss the Add User Rule window. Figure 4–4 shows a new ACL created to authorize users to update their own homePhone and homePostalAddress attributes.



Selected entries: DN regular expression

Attributes: homePhone,homePostalAddre

DN-based regular expression: ou=People, o=XYZ, c=US

Rights	Access Rules
✎	Self

Figure 4–4 Create Access Control Window

7. **In the Create Access Control Rule window, click Apply to store the new rules.**

   You can then select another set of entries and define access controls for them, as described in Step 2 on page 49.

   Configuration changes are implemented when you restart the `dsservd` daemon.

▼ To Modify an Access Control Rule

1. **Select the set of entries whose access control you want to modify, and choose Modify ACL from the Selected menu.**

   The Modify Access Control Rule window is displayed. If you double-click a rule, this window is displayed automatically.

2. **Select the rule that you want to modify, and choose Modify Access Rule from the Selected menu.**

   The Modify User Rule window is displayed. If you double-click the rule you want to modify, the Modify User Rule window is displayed automatically.

3. **Make the modification you require.**

4. **Click Apply.**

   Make any other modifications you require. When you have made and applied all the modifications, click Cancel to dismiss the Modify User Rule window.

5. **Click OK in the Modify Access Control Rule window.**

   These changes will take effect when you restart the `dsservd` daemon.

▼ To Delete an Access Control Rule

♦ **To delete all access control rules for a set of entries, select the entry set and choose Delete ACL from the Selected menu.**

You are prompted to confirm that you want to delete all access controls for the set of entries.

♦ **To delete one rule from an ACL:**

1. **Select the set of entries and choose Modify ACL from the Selected menu.**

   The Modify Access Control Rule window is displayed.

2. **Select the rule you want to delete and choose Delete User Rule from the Selected menu.**

   You are prompted to confirm that you want to delete this rule.

▼ To Reorder Access Control Rules

♦ **In the Access Control section of the main window, select the rule you want to move, and choose Move Up or Move Down from the Selected menu.**

When the rules are reordered as you require, click Apply to save the changes to the configuration file. The changes are implemented when you restart the `dsservd` daemon.

Note - The Admin Console will display a warning message if you break the convention of placing the rules from the more specific to the more general.

Configuring Logging

You can configure the following parameters for the log:

- Log directory
- Log file size
- Trace level
- Which logs to display (LDAP, web gateway, NIS, RADIUS), and the length of the display

Go to the Log section of the Admin Console main window to configure these parameters.

The log directory `/var/opt/SUNWconn/ldap/log` contains the following log files:

- `dsserv.log` -— the log file for the LDAP and NIS server
- `dsradius.log` — the log file for the RADIUS server
- `dsweb.log` — the log file for the web gateway
- `dsnmpserv.log` — the log file for the SNMP agent that monitors the `dsservd` daemon
- `dsnmprad.log` — the log file for the SNMP agent that monitors the `dsradiusd` daemon
- `dsserv_admin.log` — the log file for the Admin Console server
- `dspush.log` — the log file for the `dspushd` replication process
- `dspull.log` — the log file for the `dspulld` replication process

You can change the name of the default log directory but you cannot modify the names of the log files themselves.

By default the maximum size of a log file is 500Kb. The maximum size is the same for all log files. When this limit is reached, the log is renamed by appending a `.1` suffix, and a new log file is created. When that log reaches the size limit, it is in turn renamed. You can have up to ten files for each log.

Note - Log files can use up to 40 Mbytes of disk space, or more if you increase the default log file size. Therefore, it is necessary to clean up log files regularly.

The available trace levels are:

- None
- Light
- Medium
- Full

Backing Up and Restoring the Configuration

The Admin Console enables you to back up your directory service configuration at any time and save as many configurations as you want. You can restore any of your backed up directory service configurations.

▼ To Back Up the Configuration

1. **In the Admin Console, click the Server pull-down menu and choose Backup Config.**

   If the directory service damon `dsservd` is running, you are prompted to stop it. When it is stopped, a selector window is displayed.

2. **Use the selector window to select or create a directory for your backup.**

3. **Click Save.**

▼ To Restore a Configuration

1. **In the Admin Console, click the Server pull-down menu and select Restore Config.**

   If the directory service damon `dsservd` is running, you are prompted to stop it. When it has stopped, a selector window is displayed.

2. **Select the directory containing the backed up configuration that you want to restore.**

   The default selection is the `/etc/opt/SUNWconn/ldap/backup` directory.

3. **Click Load.**

   The directory service is restarted with the restored configuration.

Displaying Server Characteristics

You can display the characteristics of the directory server at any time, using the following command:

/opt/SUNWconn/bin/ldapsearch -h *host* **-b "" -s base 'objectclass=*'**

host is the hostname of the directory server whose characteristics you want to display. The output from this command is similar to:

```
objectclass=top
namingcontexts=O=XYZ,C=US
namingcontexts=DC=XYZ,DC=COM
subschemasubentry=cn=schema
supportedsaslmechanisms=CRAM-MD5
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=1.3.6.1.4.1.1466.20035
supportedldapversion=2
supportedldapversion=3
```

Using the DN Editor

A distinguished name (DN) is a sequence of relative distinguished names (RDNs), separated by a comma, for example o=XYZ, c=US. When you have to specify a DN in the Admin Console, you can type it directly into the field supplied, or you can construct it using the Distinguished Name Editor.

Regular Expressions in Distinguished Names

You can specify a DN that contains a regular expression to indicate a set of entries. This is useful when configuring access control, for example, but not when specifying a naming context. The Admin Console does not prevent you from entering a regular expression in any DN, but you should use wildcards only where it is appropriate. See the `regex(1F)` man page for information about regular expressions.

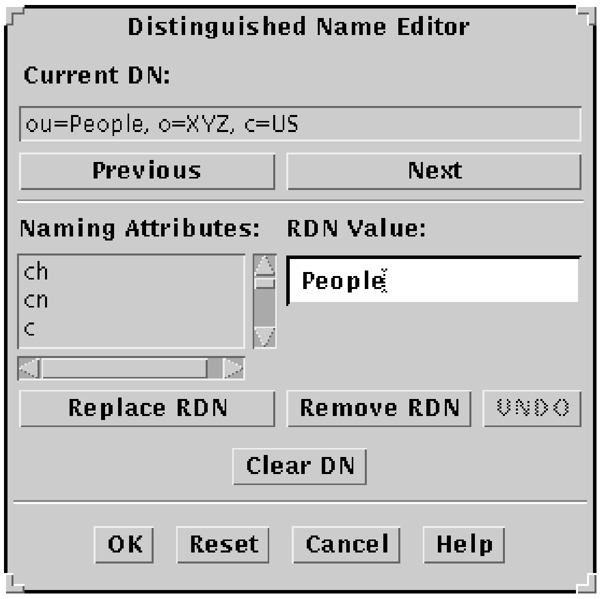
You can specify a regular expression for the distinguished name of an entry. For example, the regular expression dn="cn=Joe Smith, ou=.*, o=XYZ, c=US" specifies the set of entries for people called Joe Smith in the whole of the XYZ Corporation.

You can also use a DN-based regular expression to specify a set of values for an attribute whose values are DNs. For example, you can grant write access to a distribution list entry to any person whose DN is a value of the member attribute, using the regular expression member="dn=.*".

▼ To Create a Distinguished Name with the DN Editor

1. **Start the DN Editor by clicking the Set Button to the right of the field where you must enter a DN.**

   The DN Editor dialog box is displayed.


Distinguished Name Editor

Current DN:

ou=People, o=XYZ, c=US

Naming Attributes: ch, cn, c

RDN Value: People

Figure 4–5 Distinguished Name Editor

2. **Use the Previous and Next buttons to position the cursor where you want to add an RDN, or to highlight an RDN that you want to replace.**

   If you position the cursor between two RDNs, the dialog box has an Add RDN button. If you highlight an RDN, this button is replaced by a Replace RDN button.

3. **Select the Naming Attribute for the RDN, and type the value in the RDN value field.**

4. **Click the Add RDN button or Replace RDN button.**

5. **Click Apply to save the new DN, and click Cancel to dismiss the DN Editor dialog box.**

▼ To Modify a Distinguished Name with the DN Editor

1. **Start the DN Editor by clicking the Set Button to the right of the field containing the DN to modify.**

   The DN Editor dialog box is displayed.

2. **Use the Previous and Next buttons to position the cursor where you want to insert an RDN, or select the existing RDN that you want to replace.**

3. **Select the Attribute Type for the RDN, and type the value in the RDN value field.**

4. **Click the Modify RDN button.**

5. **Click Apply to save the new DN, and click Cancel to dismiss the DN Editor dialog box.**

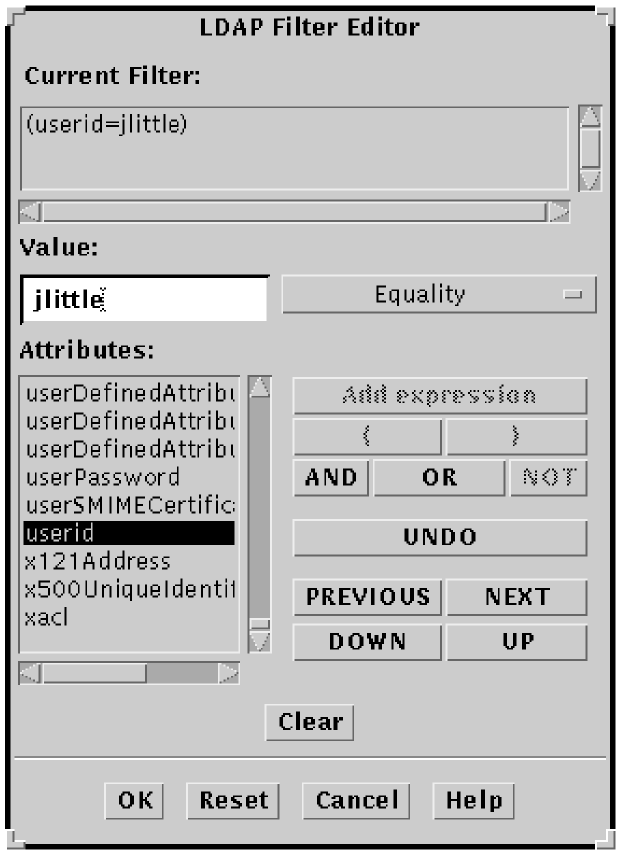
Using the Filter Editor

An LDAP filter is a way of specifying a set of entries, based on the presence of a particular attribute or attribute value. You can use an LDAP filter in an access control rule. For example, the default access control rules include a filter specifying that users can add their own DNs to the member attribute of any entry that contains the attribute joinable with a value of TRUE. This allows users to add or remove their names from distribution lists.

The Current Filter field shows the filter you are modifying, or the current state of the filter you are creating.

▼ To Create an LDAP Filter

1. **Click the Set button to the right of the field where you need to specify a filter.**
   The Filter Editor is displayed.



(userid=jlittle)

jlittle

Figure 4–6 Filter Editor Dialog Box

2. **Select an attribute from the list displayed.**

3. **Type a value in the Value field.**

4. **Select a match type from the pull-down menu.**

5. **Click AND, OR, or NOT, to indicate how this expression is used in the filter.**

6. **Click Add Expression to add the expression to the filter.**

   The Current Filter field shows the modified filter.

7. **Repeat from Step 3 on page 58 to Step 6 on page 59 until the Current Filter field shows the filter as you want it.**

8. **Click Apply to save the filter, and click Cancel to dismiss the Filter Editor dialog box.**

▼ To Modify an LDAP Filter

1. **Click the Set button to the right of the field where you need to specify a filter.**

   The Filter Editor is displayed, as shown in Figure 4–6.

   The Current Filter field shows the filter to be modified.

2. **Position the cursor where you want to add an expression, or select the expression that you want to modify.**

3. **Select an attribute from the list displayed.**

4. **Type a value in the Value field.**

5. **Select a match type from the pull-down menu.**

6. **Click AND, OR, or NOT, to indicate how this expression is used in the filter.**

7. **Click Add Expression to add the expression to the filter.**

   The Current Filter field shows the modified filter.

8. **Repeat from Step 3 on page 59 to Step 7 on page 59 until the Current Filter field shows the filter as you want it.**

9. **Click Apply to save the filter, and click Cancel to dismiss the Filter Editor dialog box.**

CHAPTER **5**

Loading and Maintaining Directory Information

This chapter describes how to use the command-line utilities supplied with Sun Directory Services to populate and maintain your database of information. To initialize the database, you can:

- Manually create the root entry and all other entries using `ldapadd` (see "Creating the Root Entry" on page 61) or Deja (see *Sun Directory Services 3.1 User's Guide*).
- Populate the directory automatically by using the `dsimport` utility (see "Populating the Directory" on page 62). If you choose this option, the root entry is created automatically.

Creating the Root Entry

You cannot add entries to your data store before you have created the root entry for the data store. The root entry is the top entry of the tree held by the data store. It identifies the data store. In Sun Directory Services, you can actually have up to four root entries that identify the data store and that correspond to the four possible data store suffixes that you can declare in the Admin Console.

To create the root entry, create a simple LDIF file containing the entry information, and add it to the database using the `ldapadd` command. An example of this procedure is given in "To Create the Root Entry for XYZ Corporation" on page 62.

You can also create the root entry manually using Deja. The procedure for adding entries using Deja is explained in *Sun Directory Services 3.1 User's Guide.*

Note - The root entry is created automatically if it does not already exist when you first load entries in the directory using the `dsimport` command.

▼ To Create the Root Entry for XYZ Corporation

1. **Create an LDIF file called `root-file` that contains:**

```
dn: o=XYZ, c=US
objectClass: organization
```

   The LDIF file format is described in detail in the `ldif(4)` man page.

2. **Add this file using `ldapadd(1)`:**

   prompt% ldapadd -c -D “cn=*admin-cn*, o= XYZ, c=US” -w *admin-pw* -f root-file

   where:

   - `-c` specifies to continue processing even if errors occur
   - `-D` introduces the distinguished name of the data store administrator. The DN must be given in quotes because it is likely to contain blank spaces.
   - `-w` introduces the administrator password
   - `-f` introduces the file holding the information to add to the database.

   If you want to avoid your password showing up in a command listing, you can omit the `-w` option. The `ldapadd` command will prompt you for your password.

   The root entry now exists.

Populating the Directory

If you do not want to create directory entries manually, you can populate the directory using the `dsimport` bulk load utility. This utility creates directory entries from any text file in which one line corresponds to one directory entry. You must create a mapping file that specifies the semantics for the information provided in each line of the input file. You might also need to create an LDAP object class and attributes that are specific to the type of information you want to store in the directory.

Refer to “Mapping Syntax and Semantics” on page 63 for information on the structure and content of a mapping file. A complete example of creating a mapping file and using `dsimport` is given in “Example: Using dsimport” on page 76.

For details on all the options of `dsimport(1m)`, refer to the man page.

The `dsimport` utility is also used during the initialization of the NIS service to import all the information stored in NIS files into the LDAP directory. When you run the `dsypinstall` script to configure Sun Directory Services as an NIS server, the NIS information available on your server is automatically added to your directory database through a call to `dsimport`. The mapping of NIS files into LDAP object classes and attributes is described in the `nis.mapping` file in the directory `/etc/opt/SUNWconn/ldap/current/mapping`. For full details on importing NIS information into the directory, see “Initializing the Sun Directory Services NIS Service” on page 88.

Note - The information mapping described in the `radius.mapping` file in the directory `/etc/opt/SUNWconn/ldap/current/mapping` is used to perform RADIUS searches in the LDAP directory, *not* to import RADIUS information.

Mapping Syntax and Semantics

The mapping syntax and semantics are designed to provide maximum flexibility so that you can easily:

- Import information from any text file into the directory
- Adapt or create the mapping for a proprietary table in your NIS environment

If this involves modifying or creating an object class with the attributes that you need, refer to “Modifying the Schema” on page 127

A mapping file is made up of a number of sections that conform to the following pattern:

```
Front-end name
 Common
 Table
  Common
  Dynamic
  Export
   Extract
   Condense
   Build
```

(continued)

(Continuation)

```
   Import
    Extract
    Condense
    Build
  Table
   Common
   Dynamic
   Export
    Extract
    Condense
    Build
   Import
    Extract
    Condense
    Build
 ...
```

The content and meaning of each section is described in "Mapping Semantics" on page 64. The syntactic rules for each section are described in "Mapping Syntax" on page 69.

Mapping Semantics

Front-end name indicates the name of the service. All the information that follows that name describes the mapping of service-specific information to LDAP object classes and attributes.

The first *Common* section immediately following the front-end name gives configuration information that applies to the front-end or service. It contains mandatory configuration variables that are required in the translation process, and optional configuration variables that are stored in the same file for convenience. In the `nis.mapping` and `radius.mapping` files, this section can be modified through the Admin Console.

The *Table* section defines mapping information for a particular type of information. The mapping information determines the object class of all entries created using that table definition. Each table definition is composed of the following sections:

- Common
- Dynamic (mandatory)
- Export
- Import

The Dynamic section is the only one that is mandatory. Without it, neither import nor export operations work. The other sections can be omitted if you do not need them. For instance, if you never intend to export information from the directory, you do not need to create an Export section.

Each section contains keywords and definitions used in the import or export process. Table 5–1provides a list of mapping keywords, the sections in which they can occur, and their purpose.

In any section, you can create variables or *tokens*, that is, private definitions, by using the following format:

```
tokenT=token definition
```

Your private definitions can use the syntax and functions described in “Condense” on page 71.

TABLE 5–1 Summary of Mapping File Keywords

Section	**Keyword**	**Mandatory/ Optional**	**Purpose**
Common	BASE_DN	Mandatory, but can be specified in the Dynamic section	Specifies a naming context. See “BASE_DN ” on page 67.
	MAP_NAME	Mandatory for an NIS table definition	Indicates the name of the NIS table corresponding to the table definition. See “MAP_NAME ” on page 67.
	PRIVATE_OBJECTCLASSES	Mandatory when object class is not unique	Used for updates on entries created from several table definitions. See “PRIVATE_OBJECTCLASSES ” on page 67.

TABLE 5–1 Summary of Mapping File Keywords *(continued)*

Section	Keyword	Mandatory/ Optional	Purpose
Dynamic	ALL_FILTER	Mandatory	Defines a filter for identifying all entries created using the table definition. See "ALL_FILTER" on page 68.
	DC_NAMING	Optional	Defines the mechanism for converting a domain name to an LDAP dc name structure. See "DC_NAMING" on page 68.
	LINE	Mandatory	Defines decomposition of input information. See "LINE" on page 68.
	MATCH_FILTER	Mandatory	Defines a filter for identifying a particular entry created using the table definition. See "MATCH_FILTER" on page 68.
Export/Build	LINE	Mandatory if the Export section exists	In export file, defines format of line composed of LDAP attributes. See "Export Section" on page 68.
	NIS_KEY	Mandatory for NIS	Identifies NIS key in export file.
	NIS_VALUE	Mandatory for NIS	Identifies NIS value in export file.
Import/Extract	LINE	Mandatory if the Import section exists	Defines decomposition of input information. See "Import Section" on page 69.

Common Section

The Common section contains definitions of variables that apply to all the entries created using that table definition but not to the entire service or front-end. For example, the Common section typically contains the naming context under which the entries are created. The naming context is specified using the `BASE_DN` keyword.

BASE_DN

The `BASE_DN` keyword specifies the naming context under the entries are to be created. The `dsimport` utility looks for this parameter in several places, in the following order:

1. Command line of `dsimport`, option `-V`
2. Dynamic section
3. Common section for the Table
4. Common section for the Front-End (at the beginning of the mapping file)

MAP_NAME

The `MAP_NAME` keyword specifies the name of the NIS map corresponding to the table definition. This keyword is used to create administrative entries for the NIS service. The directory server maintains these entries automatically.

This keyword is used also to create the naming context for the NIS entries that are created by using the generic mapping definition.

The `MAP_NAME` keyword is specific to the NIS service.

PRIVATE_OBJECTCLASSES

The `PRIVATE_OBJECTCLASSES` keyword specifies an object class when the object class and attributes derived from a table definition do not make up a complete entry. This keyword is necessary for maintaining directory entries that are created from several table definitions. This can be the case when several table definitions each create an auxiliary object class and its associated attributes.

For example, in the NIS environment, network hosts can have entries in at least three files: `/etc/bootparams`, `/etc/ethers`, `/etc/hosts`. However, each host has just one entry in the LDAP directory, with the three auxiliary object classes bootableDevice, ieee802Device, and ipHost. If the entry for the host is deleted in one of these files, the corresponding entry in the LDAP directory must not be deleted but simply updated by removing the appropriate auxiliary object class, and any attributes specific to that object class.

Dynamic Section

The Dynamic section contains equations that make it possible to dynamically build the filters required to locate relevant information.

LINE

The LINE keyword is necessary to define how the input information must be dynamically decomposed to provide the elements required in the MATCH_FILTER and ALL_FILTER definitions.

The syntax of the LINE keyword is given in “Extract” on page 70.

MATCH_FILTER

The MATCH_FILTER keyword specifies a filter that is used by the `dsimport` utility to check whether an entry already exists in the database before creating it. If it exists, the `dsimport` utility will check whether it needs to be modified.

The MATCH_FILTER keyword is also used by the directory server to respond to commands such as `ypmatch`.

ALL_FILTER

The ALL_FILTER keyword specifies a filter that is used by the `dsexport` command to regenerate the file from which the directory entries were originally created. This filter is necessary even if you do not intend to export information from the directory to regenerate the source file for that information.

The ALL_FILTER keyword is used by the directory server to retrieve from the directory all entries that belong to a given NIS table. This is because the directory server maintains a permanently up-to-date copy of the NIS tables.

The ALL_FILTER keyword is also used by the directory server to respond to commands such as `ypcat`.

DC_NAMING

The DC_NAMING keyword defines the mechanism applied to convert a domain name of the form xyz.com to an LDAP data store suffix or naming context of the form dc=xyz, dc=com. This is useful if the naming structure that you use in your directory is a domain component (dc) structure.

Export Section

The Export section provides the method for regenerating a source file from LDAP directory entries. This section is optional. When it exists, it must contain the keyword LINE. The LINE keyword in the Export section must reflect the format of a line in the original source file.

The Export section contains the following subsections:

- Condense: this optional subsection contains variable definitions that can be used in the Build subsection to build the parameters required to generate `LINE`.
- Build: Contains at least the output `LINE` definition.

In the `nis.mapping` file, the Build subsection defines the rules for constructing an NIS key / NIS value pair; it also defines the rules for generating the line in the NIS file corresponding to the LDAP directory entry.

Import Section

The Import section provides the method for translating a line in an input file into an LDAP directory entry. This section must contain a `LINE` keyword that defines how a line in the input file can be decomposed into elements that can be described by LDAP attributes. It must also contain the list of LDAP attributes that are created from a line in the input file.

The Import section contains the following subsections:

- Extract: Contains the `LINE` definition with the notation described in “Extract” on page 70.
- Condense: Contains variable definitions that can be used in the Build subsection to generate LDAP attributes and attribute values.
- Build: Provides a list of LDAP attributes, including the object class, and defines the rules for constructing the value of each LDAP attribute from the variables in the Condense section or from the parameters in the Extract section

In the `nis.mapping` file, the LINE definition in the Extract subsection specifies the rules for analyzing a line in an NIS source file into smaller units of information called NIS tokens.

Mapping Syntax

This section describes the syntax of the variables or tokens that you can create in each section of a table definition.

The mapping syntax is described using examples from the `nis.mapping` file.

Common

The variables defined in the Common section other than the keywords listed in Table 5–1 must follow this syntax:

```
variable-name=value
```

Variables defined in the Common section contain static configuration information.

Dynamic

The variables defined in the Dynamic section other than the keywords listed in Table 5–1follow the same syntax as the variables defined in the Common section. However, their values are supplied in the input to the utility (such as `dsimport` or `dsexport`) that uses the mapping file during its execution.

Extract

The variables defined in the Extract section define the rules for decomposing input information into smaller units of information, called tokens, that can be directly mapped onto LDAP attributes, or that require simple processing in order to be mapped onto LDAP attributes.

The syntax of a variable that defines a decomposition into tokens is:

```
VARIABLE => $element1 separator $element2 [separator $elementn...|| ...]
```

The separator between tokens is the separator expected in the input information. It could be white space, a comma, a colon or any other character. However, one space in the line definition will match any number of spaces or tabs in the actual input information. You can specify several alternatives for the decomposition, by using two pipe symbols (`||`) to introduce each alternative rule.

The conversion process examines the rules in the order in which they are specified, and applies the first one that matches the information it was given in input.

For example, in `nis.mapping`, the following definition extracts tokens from a line in the `bootparams` file:

```
LINE =>$ipHostNameT $parametersT
```

The `hosts` file provides a slightly more complex example:

```
 LINE =>$dummy $ipHostNumberT $ipHostNameT $allIpHostAliasesT*#$descriptionT*||\
          $dummy $ipHostNumberT $ipHostNameT $allIpHostAliasesT||\
          $dummy $ipHostNumberT $ipHostNameT
```

In these examples, the tokens parametersT and allIpHostAliasesT require further processing before they can be mapped onto LDAP attributes. The processing required is defined in the Condense section.

Condense

The Condense section contains variables that define operations on tokens resulting from the Extract section, or any previously defined variable in the table definition.

It simplifies the attribute value definitions given in the Build section.

Variables defined in the Condense section can contain:

- A token specified in the same table section
- A configuration variable specified in the Common section for the same NIS table, or in the Common section that applies to all NIS tables
- A constant expression
- A function: `exclude`, `getrdn`, `split`, `instances2string`, `string2instances`, or `trim`. A variable can contain just one function. If you want to use several functions on the same information, you must create intermediate variables.

Variables in the Condense section can be made up of several alternative rules. The conversion process applies the first rule that matches the input information. The rules must be separated by two pipe symbols, and must all be part of the same expression. For example, the following expression is permitted:

```
fifi=$parameter1 - $parameter2 || $parameter1 || juju
```

whereas, the following expression is not:

```
fifi=$parameter1 - $parameter2
fifi=$parameter1
fifi=juju
```

You can define any number of variables in the Condense section. The order in which they are listed is important if you create dependencies between them. For instance, you can have:

```
fifi=$parameter1 - $parameter2 || $parameter1 || juju
riri=$fifi - $parameterA
loulou=$fifi - $parameterB
```

split Function

The syntax of the `split` function is as follows:

```
variableA=split(what, "separator", "add_prefix", "add_suffix", order)
```

where:

variableA identifies the variable

what identifies the unit of information, variable or parameter, to which the operation applies

separator indicates where to split the information. This value must be specified between quotes because it could contain a space.

add_prefix specifies a prefix to add to each item resulting from the split. This value must be specified between quotes because it could contain a space.

add_suffix specifies a suffix to add to each item resulting from the split. This value must be specified between quotes because it could contain a space.

order specifies the order in which the items resulting from the split are to be presented. The possible values for this parameter are left2right or right2left.

For example, in the `nis.mapping` file, the following variable definition is used to split an NIS domain name into a sequence of LDAP domain component attributes:

```
DC_NAMING=split($DOMAIN_NAME, ".", "dc=", ",", left2right)
```

If the domain name specified is eng.europe.xyz.com, the resulting expression is:

dc=eng, dc=europe, dc=xyz, dc=com.

string2instances Function

The `string2instances` function breaks down a specified string into instances. The syntax for this operation is:

```
variableA=string2instances("string", "separator")
```

where:

variableA identifies the variable

string identifies the unit of information, variable or parameter, to which the operation applies. This value must be specified between quotes because it could contain a space.

separator indicates where to split the information into instances. This value must be specified between quotes because it could contain a space.

For example, in `nis.mapping`, the following definition in the Condense section of the `bootparams` file breaks down a string of parameters into separate instances:

```
bootParameterT=string2instances($parametersT," ")
```

The `string2instances` function is also used to specify the inheritance tree for an object class. For example, if the object class of an entry created using a particular mapping definition is organizationalPerson, the Condense section of the mapping definition must contain the line:

```
objectClassT=string2instances("top person organizationalPerson", " ")
```

instances2string Function

The `instances2string` function combines several instances into a single string. The syntax for this operation is:

```
variableA=instances2string(what, "separator")
```

where:

variableA identifies the variable

what is a variable that has a number of instances

separator marks the separation between the elements of the string. This value must be specified between quotes because it could be a space.

For example, you could use the following variable to find the list of names and alias names for a given machine:

```
NameList=instances2string($cn, " ")
```

If the cn attribute has the values camembert, Cam, Bertie, the resulting string would be:

camembert Cam Bertie

trim Function

The `trim` function removes any unnecessary white space surrounding a parameter. The syntax for the `trim` operation is:

```
variableA=trim(parameter)
```

where:

variableA identifies the variable

parameter is the item from which white space must be removed

For example, if you decompose an alias list into its constituent members, you could define the following variables:

```
aliasMember=string2instances($aliasList, ",")
trimAliasMember=trim($aliasMember)
```

Each aliasMember parameter resulting from the `string2instances` operation is processed to remove any white spaces.

getrdn Function

The `getrdn` function returns the naming attribute of an entry, that is the attribute used in the entry's RDN. The syntax for the `getrdn` operation is:

```
variableA=getrdn()
```

Note - The `getrdn` function can only be used in variables in the Condense section.

For example, the cn attribute of a machine has the values camembert, Cam, Bertie, but the actual system name of the machine, used in the RDN is camembert. For example, you could create the following variable:

```
HostName=getrdn()
```

The `getrdn` function returns the name camembert.

Note - The `getrdn` function is case-sensitive.

exclude Function

The `exclude` function removes a value from a list or a string. The syntax for this operation is:

```
variableA=exclude(string, exclude-value, "separator")
```

where:

variableA identifies the variable

string identifies the list or string

exclude-value is the value to exclude

separator marks the separation between the elements of the list or string. This value must be specified between quotes because it could be a space.

For example, to obtain the list of aliases for a machine, you need to exclude the canonical name from the list of names. You could create the following variables:

```
NameList=instances2string($cn, " ")
HostName=getrdn()
HostAliases=exclude(NameList, HostName, " ")
```

In `nis.mapping`, the Condense section of the `hosts` mapping definition contains:

```
ipHostAliasesLineT=exclude($allIpHostAliasesT,$ipHostNameT, " ")
```

This definition excludes the ipHostName from the list of alias names for the host.

Build

The Build subsection contains a list of LDAP attributes and the definitions of their values. It must contain at least all mandatory attributes for an object class, and the DN. If the DN definition is missing from the Build section, the entries cannot be created in the directory.

Note - You do not need to specify a DN definition in the Build sections of the `radius.mapping` file because this file is not used to import entries into the directory.

Attribute value definitions can be made up of:

- A variable or keyword specified in any of the sections of the table definition

- A configuration variable specified in the Common section that applies to all the tables in the Front-End section
- A constant expression
- A concatenation of any of the above

The syntax of an LDAP attribute and its associated value definition in the Build section is as follows:

```
LDAPattribute=attributeValueDefinition
```

For example, if you wanted to create an entry for a mail alias, and use the LDAP attribute rfc822mailMember to store the names of alias members, your mapping would contain the following definitions:

```
Condense:
 aliasMember=string2instances($aliasList, ",")
 trimAliasMember=trim($aliasMember)
...
Build:
 rfc822mailMember=$trimAliasMember
...
```

Example: Using dsimport

This example describes how to create a mapping file, and use `dsimport` to perform a bulk load of information stored in a text file.

Input File

The file containing the information to import into the LDAP directory could be an extract from a corporate online directory service that provides basic information about employees.

This file is shown in Code Example 5–1.

CODE EXAMPLE 5–1 Example of Input File

```
Rob Green, rgreen@london.XYZ.com, phone x 44 1234, marketing communications
manager
Jean White, jwhite@london.XYZ.com, phone x44 1123, documentation manager
Susan Brown, sbrown@london, phone (44) 123 45 67 00, technical writer
Karen Gray, kgray@london, tel (44) 123 45 67 01, engineering project manager
Steve Black, sblack@eng, x44 1122, software development engineer
Felipe Diaz Gonzalez, fdgonzalez@eng, x41 2233, software development
engineer
Anne Marie de la Victoire, amvictoire@paris.xyz.com, x33 3344, software
support engineer
DURAND Pierre, pdurand@paris, tel 33 1133, software support engineer
```

In this file, there is one employee definition per line. On each line the information is ordered as follows:

- Name
- E-mail address
- Telephone number
- Job description

The level of information is not always consistent for the various employees: the e-mail address is not always fully qualified, the telephone number is not always a complete telephone number but an extension, and in one case the last name is given before the first name.

If you want a consistent level of information for the entries that will be created in the LDAP directory, you must either make the necessary corrections in the source file, or make them after the import operation using the Deja tool.

Mapping File

The intention of the directory administrator is to create all employee entries under the naming context ou=People, o=XYZ, c=US. The token that specifies this in the mapping file is the BASE_DN token in the Common section.

The information will be imported just once. Therefore, it is not necessary to define an Export section in the mapping file. The Dynamic section is mandatory. The object class definition in the Condense section is also mandatory.

The mapping file created by the directory administrator is shown in Table 5–2.

TABLE 5–2 Example of Mapping File

```
# This example file for dsimport contains employee entries
# to be created as inetOrgPerson objects,in the LDAP directory.

Front-End: EXAMPLE

Table: People

 Common:
  BASE_DN=ou=People, o=XYZ, c=US

 Dynamic:
  LINE => $cn, $rest
  MATCH_FILTER=(&(objectClass=inetOrgPerson)(cn=$cn))
  ALL_FILTER=(&(objectClass=inetOrgPerson)(cn=*))

 Import:
  Extract:
   LINE => $fn $ln, $mail, $telephoneNumber, $description
  Condense:
   cn=$fn $ln
   snT=$ln
   objectClassT=string2instances("top person
organizationalPerson inetOrgPerson", " ")
  Build:
   dn=cn=$cn, $BASE_DN
   sn=$snT
   mail=$mail
   telephoneNumber=$telephoneNumber
   description=$description
   objectClass=$objectClassT
```

The Condense section contains the inheritance tree for the inetOrgPerson object class.

The Build section contains all the mandatory attributes pertaining to or inherited by the inetOrgPerson object class. It also contains the optional attributes pertaining to this object class that the directory administrator required.

Running dsimport

To import the file described in "Input File" on page 76 sing the mapping file described in "Mapping File" on page 77, you can use `dsimport` with the following arguments:

```
# dsimport -h hostname -D cn=admin,o=xyz,c=us -w secret -m mapping.file
-f EXAMPLE -t People input.file
```

(continued)

where:

- *hostname* is the name of the host that holds the directory data store
- cn=admin,o=xyz,c=us is by default the distinguished name of the directory administrator
- secret represents the password of the directory administarator
- *mapping.file* contains the mapping for the input information
- EXAMPLE is the front-end specified in the mapping file
- People is the table specified in the mapping file
- *input.file* contains the input information

It is not strictly necessary to specify the DN and password of the administrator on the command line. If you omit these parameters, `dsimport` will read them from the `dsserv.conf` file. The advantage is that the DN and password of the administrator will not be displayed by the `ps` command.

After running this command, the following message is displayed:

```
Lines read: 9, processed: 8  Entries: added 10, modified 0, deleted 0, errors 0
```

The line count includes blank lines. The number of entries created is greater than the number of lines in the file because the `dsimport` command automatically creates the root entry, in this example o=xyz, c=US.

Data Management

Once you have populated your database with the information you need to run the directory service, you need to maintain that directory information by adding, modifying, or deleting entries. This section summarizes the command line utilities that you can use to maintain directory information.

For information on performing data management tasks from a graphical user interface, refer to *Sun Directory Services 3.1 User's Guide*.

Adding Entries

You can add an entry to the directory using `ldapadd(1)`. You can specify a single entry on the command line, or you can specify one or more entries in a file. See the `ldapmodify(1)` man page (`ldapadd` is a particular configuration of `ldapmodify`) for details of how to use `ldapadd`.

You can use `dsimport` with the `-n` option to create an LDAP Data Interchange Format (LDIF) file suitable for use with `ldapadd`. You can also create your own LDIF file manually, and use the `ldifcheck(1m)` command to validate it. The format of LDIF files is described in the `ldif(4)` man page.

Modifying Entries

You can modify an entry in the directory using the `ldapmodify` or `ldapmodrdn` command.

Use `ldapmodify(1)`:

- to modify the attributes in a single entry, by specifying the modification on the command line
- to modify multiple entries, by specifying a file containing entry modification information

See the `ldapmodify(1)` man page for details of how to use `ldapmodify`. You can use `dsimport` with the `-n` option to create an LDIF file suitable for use with `ldapmodify`.

Use `ldapmodrdn(1)` to modify the naming attribute of an entry. Changing the naming attribute changes the distinguished name of the entry. See the `ldapmodrdn(1)` man page for details of how to use `ldapmodrdn`.

Deleting Entries

You can delete an entry in the directory using `ldapdelete(1)`. For details see the `ldapdelete(1)` man page.

Directory Maintenance

This section describes the tasks that you can perform on a regular basis to save space and to maintain Sun Directory Services performance.

Regenerating Indexes

You can regenerate the index database for a specific data store or for all data stores on the server using the `dsidxgen` command. Although the index files are automatically updated, regenerating the index database is a useful operation because it frees up disk space. Regenerating indexes helps improve performance on search operations.

For details, see the `dsidxgen(1m)` man page.

Regenerating the Database

When changes have been made to the directory database, the use of disk space is not optimal. To improve the use of disk space, you can regenerate the database by performing a backup followed by a restore.

You can back up the directory database in text format using the `ldbmcat` command. This command converts an LDBM database to the LDIF described in the `ldif(1m)` man page. For details, see the `ldbmcat(1m)` man page.

You can restore the directory database from the LDIF file created during a previous backup using the `ldif2ldbm` command. For details, see the `ldif2ldbm(1m)` man page.

For example, stop the directory server, then use the following sequence of commands to regenerate the directory database:

```
# ldbmcat id2entry.dbb > /usr/tmp/filename
# rm /var/SUNWconn/ldap/dbm/*
# ldif2ldbm -j 10 -i /usr/tmp/filename
```

Note - You must stop the directory server before you regenerate the directory database.

If your directory server is also an NIS server, you must rebuild the NIS maps using the `dsypinstall(1m)` script. You can then restart the directory server.

Checking Log Files

The log directory, by default `/var/opt/SUNWconn/ldap/log`contains eight log files, `dsserv.log`, `dsradius.log`, `dsweb.log`, `dsnmpserv.log`, `dsnmprad.log`, `dsserv_admin.log`, `dspush.log`, `dspull.log`. When a log file reaches its maximum size, by default 500Kbytes, another one is created, with a `.1` suffix. When this one in turn reaches the maximum size, another one is created with a `.2` suffix, and so on up to `.9`. This means that you can have up to 40 log files of 500 Kbytes each.

Because the log file mechanism can use a lot of disk space, it is good practice to delete log files that are no longer of any use to you.

Using `dejasync`

Whenever you modify the configuration of the NIS service or of the RADIUS service, or the mapping files for these services, respectively `nis.mapping` and `radius.mapping` under `/etc/opt/SUNWconn/ldap/current`, you must run the `dejasync` command so that these modifications are taken into account by the Deja tool. The `dejasync` command modifies the `Deja.properties` file.

You must also run `dejasync` when you initialize the NIS service so that you can use Deja to manage NIS entries.

CHAPTER **6**

Using the Directory as an NIS Server

The Network Information System (NIS) provides a robust and reliable naming service. However, it has certain limitations that may justify migrating to the NIS server provided with the Sun Directory Services.

This chapter explains how to migrate from an NIS naming service to Sun Directory Services. It describes the Sun Directory Services daemons that provide the NIS services and the NIS-to-LDAP information mapping.

Migrating from NIS to Sun Directory Services

The NIS server provided with the Sun Directory Services overcomes some of the limitations of a classic NIS naming service, namely:

- A classic NIS server propagates whole tables, not just updates
- A classic NIS server cannot handle very large tables, that is, tables that hold over 50,000 to 60,000 entries

If you have a well established NIS environment, the best way to manage the transition without disrupting your naming service is outlined below in Figure 6–1, Figure 6–2 and Figure 6–3.

Figure 6–1 shows a pure NIS environment with NIS requests from clients handled by the closest NIS server on the network. The synchronization of information held on master and slave servers is handled through NIS propagation of tables.

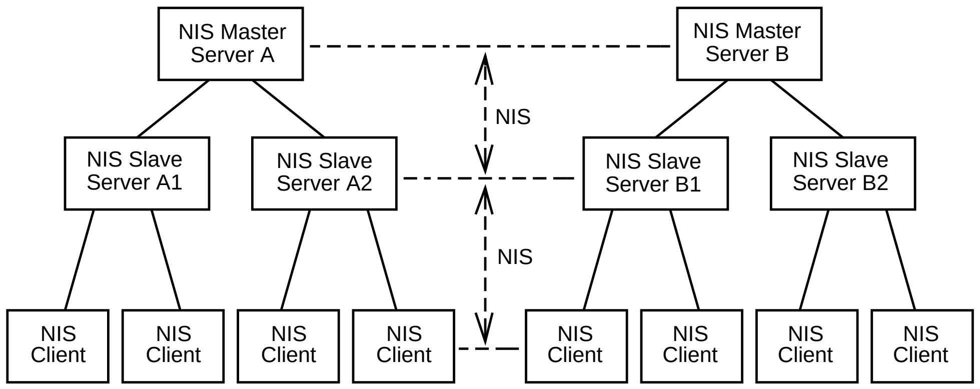


Figure 6–1 Pure NIS Environment

Figure 6–2 shows a progressive replacement of NIS slave servers by Sun Directory servers. Replacing slave servers before master servers is the recommended approach because the change will affect just some clients and not the entire network. This gives you the opportunity to adapt your procedures to the new tools. You can then apply these procedures when you do decide to migrate your NIS master servers to Sun Directory Services.

At this stage, your local slave servers also support LDAP clients. You must still use NIS for replication.

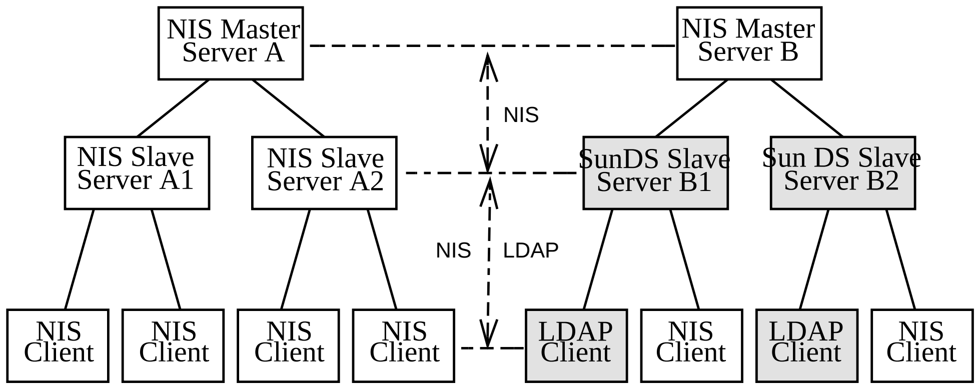


Figure 6–2 Mixed NIS - Sun Directory Services Environment

Figure 6–3 shows an environment where all legacy NIS servers have been replaced by Sun Directory Servers. Your servers support both NIS and LDAP requests, and you should use LDAP replication between master servers and slave servers.

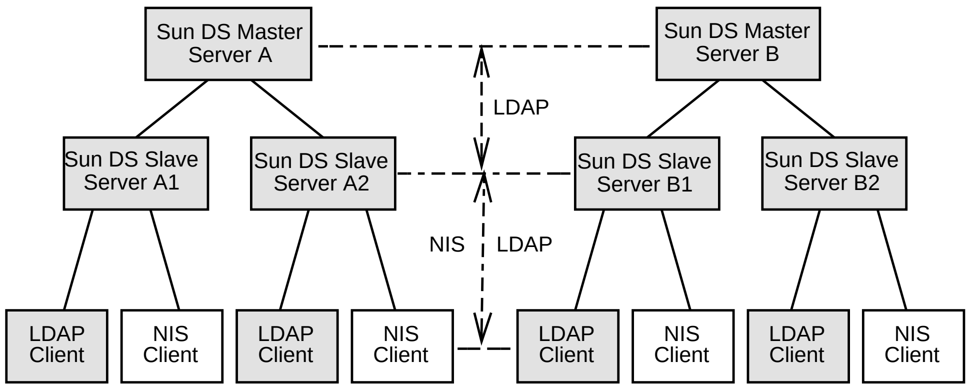


Figure 6–3 Sun Directory Services Environment

NIS-LDAP Functional Equivalence

Using the Sun Directory Services, you can perform the same level of service as with NIS. Table 6–1 shows how the standard NIS functions are preserved in Sun Directory Services.

TABLE 6–1 LDAP-to-NIS Functional Equivalence

Sun Directory Services	Classic NIS	NIS Service
dsservd	ypserv	Server process that responds to NIS requests
dsyppasswdd	rpc.yppasswdd	Daemon used to modify the NIS `passwd` tables
dsypxfrd	ypxfrd	Daemon used to transfer NIS tables to synchronize master and slave servers
dsyppush	yppush	Command used on the master server to propagate updates to slave servers
dsypxfr	ypxfr	Command used on the slave NIS server to request an update from the NIS master
dsmakedbm	makedbm	Process that builds NIS tables from standard files

TABLE 6–1 LDAP-to-NIS Functional Equivalence *(continued)*

Sun Directory Services	Classic NIS	NIS Service
dsypinit	ypinit	Process that initializes NIS clients, NIS master servers and NIS slave servers
dsyprsvd	rpc.nisd_resolv	DNS access

dsservd

The `dsservd` daemon receives and responds to NIS requests in the same way as the NIS daemon `ypserv`. It supports NIS natively. It does not convert NIS requests to LDAP requests, and LDAP responses to NIS responses.

The `dsservd` daemon also maintains an up-to-date copy of the NIS tables so that propagation requests can be handled rapidly. It converts directory information to NIS tables based on the mapping information provided in the `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping` file.

For details of how to configure the `dsservd` daemon, refer to the `dsservd(8)` man page.

Note - The `dsservd` daemon is also the LDAP server.

dsyppasswdd

The `dsyppasswdd` daemon manages changes to the password tables stored in the LDAP directory. It runs on the master server and responds to requests from users who invoke the `passwd` command to change their password, full name (in the gecos field) or shell. The `dsyppasswdd` daemon makes updates to the LDAP database, and also to the NIS `passwd` file. It also updates the `shadow` file when there is one.

For details of how to configure the `dsyppasswdd` daemon, refer to the `dsyppasswdd(8)` man page.

dsypxfrd

The `dsypxfrd` daemon manages the propagation of NIS tables from master servers to slave servers in the legacy NIS environment. It does not manage LDAP replication of NIS information. This process runs only on master servers.

The `dsypxfrd` daemon operates in the same way as the `ypxfrd` daemon in the NIS environment. It waits for a request from the `dsypxfr` process running on the slave server, then pushes the tables requested. To speed up the process the `dsservd` daemon holds an up-to-date copy of all NIS tables. This means that the NIS tables do not need to be built from the information held in the LDAP directory before they can be transferred.

For details of how to configure `dsypxfrd`, refer to the `dsypxfrd(1m)` man page.

dsyppush

The `dsyppush` command is used on a master server to propagate updates to NIS slave servers. It is called by the NIS Makefile in `/var/yp`. Also, when the Autopush option is set to On in the Admin Console, it is automatically activated when modifications are made to the directory.

For details on `dsyppush`, refer to the `dsyppush(1m)` man page.

dsypxfr

The `dsypxfr` command is used on a slave server to request updates from the NIS master server. It calls the `dsypxfrd` daemon that runs on the master server to update the local database. This update is performed using standard NIS propagation.

For details of how to use `dsypxfr`, refer to the `dsypxfr(1m)` man page.

dsmakedbm

The `dsmakedbm` command builds the NIS tables from the information held in the NIS source files. It is slightly different from the standard NIS `makedbm` command because it calls the `dsimport` utility to create NIS entries in the directory.

dsypinit

The `dsypinit` command is used to declare and initialize an NIS server as a master server or slave server. This command can also be used instead of the NIS standard `ypinit` command to initialize NIS clients.

For details of how to use `dsypinit`, refer to the `dsypinit(1m)` man page.

Note - Do not use `dsypinit` to initialize the NIS service. Use `dsypinstall`, as described in "To Initialize the NIS Service" on page 88.

---

dsyprsvd

The `dsyprsvd` daemon contacts the DNS server when the requested host information cannot be found in the NIS entries in the directory. This daemon runs only when the NIS server is configured to run in DNS interoperability mode. The NIS initialization script, `dsypinstall`, prompts you for this information.

For more information on `dsyprsvd`, refer to the `dsyprsvd(1m)` man page.

Initializing the Sun Directory Services NIS Service

The NIS service provided by Sun Directory Services preserves your existing NIS environment. It takes into account the customizations you may have made to the standard NIS environment. The initialization process modifies your existing NIS `Makefile` so that the make process calls `dsmakedbm` and `dsimport` instead of the standard NIS `makedbm` command. It also changes the `yppush` command to use `dsyppush`.

The NIS service provided by Sun Directory Services allows you to map NIS domain names to the LDAP directory tree by using a domain component (dc) naming structure. For example, information relative to the domain sales.XYZ.com can be stored in the LDAP directory under the subtree dc=sales, dc=XYZ, dc=com.

▼ To Initialize the NIS Service

1. **Install and license Sun Directory Services.**

   These tasks are described in the installation instructions.

   When the package installation is complete, a message indicates that you must run the `dsypinstall` script to initialize the NIS service. However, before you do, you must perform the configuration steps described in this procedure.

2. **Start the Admin Console.**

   This procedure is described in "Displaying the Admin Console" on page 34.

3. **Create the naming contexts that you will use to store NIS information.**

   You must create a naming context that is an appropriate domain component (DC) tree suffix for the domain that the server will manage. The initialization script will derive the naming contexts for NIS entries from the NIS domain name that you specify. For example, if you specify the domain name sales.XYZ.com, you must have a naming context of the form dc=XYZ, dc=com.

   Alternatively, you can modify the `nis.mapping` file to specify a different naming structure from the default DC structure derived from the NIS domain name. For information, refer to “Configuring Naming Contexts” on page 105.

   To create a naming context, see “To Create a Data Store” on page 44.

4. **Backup your current NIS files and database.**

5. **As root, run `dsypinstall`:**

   ```
   # /opt/SUNWconn/sbin/dsypinstall
   ```

   The `dsypinstall` script assumes that your `Makefile` is located in `/var/yp`. It also assumes that the source files for NIS tables are all located in the directory that you specify when prompted, except for the `aliases` file which is assumed to be in `/etc/mail`.

   You are prompted to enter the name of the NIS domain managed by the server and to specify whether you want to enable DNS interoperability.

   When the `dsypinstall` script has successfully finished, the NIS server is initialized and the LDAP directory database contains the information extracted from the NIS tables.

6. **Check the NIS status displayed in the Admin Console.**

   If the Admin Console was running before you ran the `dsypinstall` script, in the Status section, click Check Status to display the current status.

   In the Status section, the NIS service should be listed as Running, and automatic restart should be Enabled. In the NIS section, the NIS functional role should be shown as "NIS master server" or "NIS slave server" depending on how you declared the server during the `dsypinstall` initialization. The NIS section also shows the list of all the NIS maps supported by the server.

7. **Run `dejasync` on the server. As `root` type:**

   ```
   # /opt/SUNWconn/ldap/sbin/dejasync
   ```

For details on the options of the `dejasync(1m)` command, refer to the man page. You must run `dejasync` if you want to use the Deja tool to modify NIS entries in the directory.

For details on how NIS information is imported and stored in the LDAP directory, refer to "NIS Information in the LDAP Directory" on page 92.

Configuring the NIS Service

From the Admin Console, you can configure the following parameters for your NIS service:

- The NIS domain name
- The distinguished name of the subtree that will hold NIS administrative entries. These entries are maintained automatically by the server.
- Whether or not you want to use standard NIS replication
- Whether or not you want to include all directory entries: this option enables you to regenerate NIS maps to include entries present in the directory before the NIS service was initialized for the first time. These entries are not included in the import operation which takes place when the NIS service is initialized. You also need to use this option to regenerate an NIS map in the directory when the mapping configuration defined in the `nis.mapping` file has been modified.
- Which maps are supported on the server
- Whether you want updated information to be pushed automatically to slave servers when changes are made to NIS entries on this server, and the delay. To use this option, you must enable standard NIS replication.

To configure these parameters, go to the NIS section of the Admin Console main window.

Updating NIS Tables

Once you have populated the directory in this way for the first time, you have two options for data maintenance:

- Make updates to the NIS files, and run `make` in the `/var/yp` directory
- Make updates to the entries in the directory, for example using the Deja tool

Making updates to the entries in the directory is the most efficient method of maintaining NIS information, however the directory content will no longer be synchronized with the contents of the NIS files. If you want to resynchronize, you can export the NIS entries held in the directory to the corresponding NIS files by using `dsexport`. For details, see the `dsexport(1m)` man page.

The NIS maps are regenerated periodically from the entries stored in the directory. However, you can rebuild a map at any time from the Admin Console using the Regenerate Map function. This function will only take into account the entries stored in the directory, *not* the NIS source files. To regenerate an NIS map from the directory entries:

1. **In the NIS section of the Admin Console, highlight a map in the map list.**

2. **Optionally, set the Include All Directory Entries option to yes.**

   This is useful if you want to regenerate a map immediately following the initialization of the NIS service, or if you have changed the mapping definition for that particular map in the `nis.mapping` file.

3. **Click the Regenerate Map button.**

Note - When you include in the NIS maps maintained by the directory server the entries that were present before you initialized the NIS service, you must ensure that these entries do not create a security risk for your NIS service. For example, because users have write access to their own directory entry, it is possible for a user to change the uid attribute to become `root` user.

Propagating NIS Tables

There are two methods of propagating NIS tables between master servers and slave servers. Between two Sun Directory Services servers, choose LDAP replication. Between a Sun Directory Services server and a legacy NIS server, you must use standard NIS replication.

Note - Do not use both LDAP replication and standard NIS replication on the same subtrees or individual entries. As a general rule, use only one replication method between two servers.

Standard NIS Replication

If you make updates to your NIS files rather than to NIS entries in the directory, when you run `make` to rebuild the NIS tables, the `dsyppush` command is automatically executed.

LDAP Replication

If you make updates to NIS entries in the directory, you can enable automatic pushes of all maps to take place using the Admin Console. Alternatively, you can use the Synchronize Replicas button at any time to push just the maps you select.

For information on configuring LDAP replication, see Chapter 9.

NIS Information in the LDAP Directory

The NIS information imported into the LDAP directory is extracted from standard NIS files, and from customers' proprietary files. Sun Directory Services provides a mapping of standard NIS information to LDAP attributes. The default `nis.mapping` file explicitly describes the information mapping for the following NIS files:

- `aliases`
- `bootparams`
- `ethers`
- `group`
- `hosts`
- `netgroup`
- `networks`
- `passwd`
- `protocols`
- `rpc`
- `ypservers`

Mapping definitions for the following standard NIS files, and for proprietary NIS files listed in the NIS makefile, are not present in the default `nis.mapping` file. Mapping definitions for these files are created automatically when you run `dsypinstall` to initialize the NIS service:

- `auto_home`
- `auto_master`
- `netid`
- `netmasks`
- `publickey`
- `services`
- `timezone`

The mapping definition for these files and any proprietary files is generic, and is identical for every map. The only difference is the map name.

Table 6–2 gives a summary of the type and location of LDAP entries for each NIS map. For details on configuring the mapping definition in the `nis.mapping` file, refer to “NIS Information Mapping” on page 95.

TABLE 6–2 NIS/LDAP Mapping Summary

NIS Source File	NIS Maps	Object Class of Entries Created	Naming Context of entries created
`/etc/mail/aliases`	mail.aliases, mail.byaddr	nisMailAlias	ou=Aliases, ou=Services
`/etc/bootparams`	bootparams	bootableDevice	ou=Hosts, ou=Services
`/etc/ethers`	ethers.byaddr, ethers.byname	ieee802Device	ou=Hosts, ou=Services
`/etc/group`	group,byname, group.bygid	posixGroup	ou=Group, ou=Services
`/etc/hosts`	hosts.byaddr, hosts.byname	ipHost	ou=Hosts, ou=Services
`/etc/netgroup`	netgroup	nisNetGroup	ou=Netgroup, ou=Services
	netgroup.byhost	nisObject	ou=netgroup.byhost, ou=Services
	netgroup.byuser	nisObject	ou=netgroup.byuser, ou=Services
`/etc/networks`	networks.byname, networks.byaddr	ipNetwork	ou=Networks, ou=Services
`/etc/passwd`	passwd.byname, passwd.byuid	posixAccount	ou=People
`/etc/protocols`	protocols.byname, protocols.bynumber	ipProtocol	ou=Protocols, ou=Services
`/etc/rpc`	rpc.bynumber	oncRpc	ou=rpc, ou=Services

TABLE 6–2 NIS/LDAP Mapping Summary *(continued)*

`/var/yp/binding/`*domainName*	ypservers	sunNisServer	ou=ypservers, ou=Services
All other maps listed in `makefile`	*mapName*	nisObject	ou=*mapName*, ou=Services

NIS Naming Contexts

During the NIS initialization phase, the information imported into the LDAP directory is organized according to the tree structure shown in Figure 6–4. In this figure, ou=*mapName* represents any map not defined in the default `nis.mapping` file, for example `services.byname`.

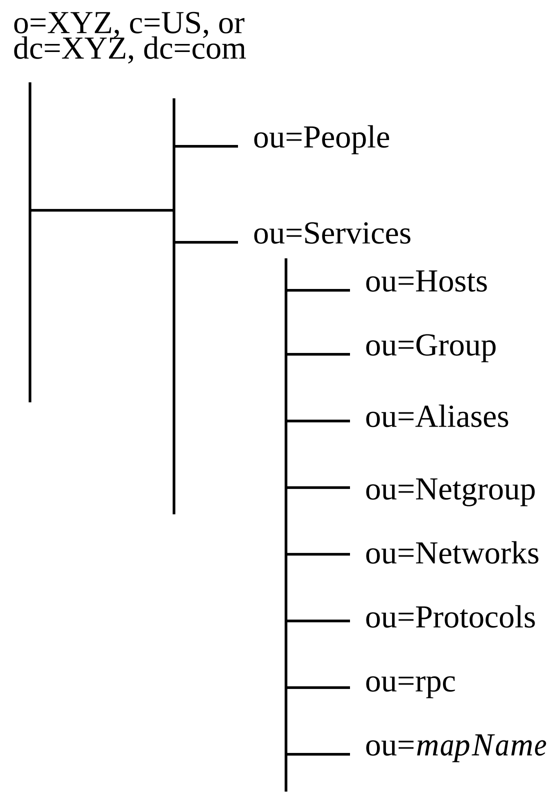


Figure 6–4 Tree Structure for NIS Information

This tree structure can be configured to suit your needs by modifying the mapping file `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`.

NIS Information Mapping

The mapping of entries in NIS tables to LDAP attributes is defined in the file `/etc/opt/SUNWconn/ldap/current/mapping/nis.mapping`. This file is divided into sections that contain mapping information for each NIS table. At the beginning of the file, there is a Common section that contains configuration information that applies to every table.

The `nis.mapping` file is used by the following components:

- The `dsimport` utility, to extract information from NIS files and convert it to LDAP directory entries.
- The `dsexport` utility, to extract information from LDAP entries and restore the NIS files in their original format.
- The `dsservd` daemon, to convert LDAP entries to NIS table entries. The `dsservd` daemon maintains up-to-date copies of all NIS tables held in the LDAP database to respond to synchronization requests from legacy NIS slaves.
- The `dsypxfrd` daemon, to check that the tables specified in synchronization requests are supported by the mapping file.

The mapping definitions in the `nis.mapping` file specify the following:

- The object class and attributes used to create an entry in the directory
- The naming context under which the entries will be created
- Case-sensitivity of the directory server for searches on the entries

Object Class and Attributes

The object class and attributes of an entry created from a NIS file are listed in the Import/Build section of the mapping definition for a given table. The values assigned to the LDAP attributes listed in the Build section are taken from the tokens specified in the other sections of the table definition.

A generic mapping is applied to all files listed in the NIS `Makefile` that do not have a specific mapping definition in the `nis.mapping` file. This mapping extracts the NIS key and NIS value information from any file and creates an entry based on the following generic NISobject object class, as follows:

cn	nisKey (case-insensitive on LDAP searches)
sunNisKey	nisKey
nisMapEntry	nisValue
nisMapName	map name declared in makefile
objectClass	top
	nisSunObject

The object classes and attributes created for all the standard NIS tables that are described in the mapping file are given below. To avoid repetition, the `aliases` file mapping is the only one that is commented in detail.

aliases

An entry in the `aliases` file is usually of the form:

```
aliasname: mailaddress1, mailaddress2, mailaddress3...
```

The mapping of this line in the `nis.mapping` file is shown below:

```
Table: mail.aliases
...
Import:
 Extract:
  LINE =>$aliasNameT:$aliasListT
 Condense:
  rfc822mailMembersT=string2instances($aliasListT,",")
  trimrfc822mailMembersT=trim($rfc822mailMembersT)
  objectClassT=string2instances("top nisMailAlias", " ")
 Build:
  dn=cn=$aliasNameT,$BASE_DN
  cn=$aliasNameT
  rfc822mailMember=$trimrfc822mailMembersT
  objectClass=$objectClassT
```

In the Extract section, the keyword `LINE` shows the initial decomposition of a line in the `aliases` file. If in your `aliases` file you use a punctuation mark other than a colon to separate the alias name from the alias list, you must change the `LINE` definition to match your file format.

In the Condense section, the token definition for rfc822mailMembersT provides a further decomposition of the alias list. If in your `aliases` file you use a punctuation mark other than a comma to separate the elements in the alias list, you must change the token definition to match your file format.

The token definition for trimrfc822mailMemberT removes any unnecessary spaces from the result produced by the preceding `string2instances` operation on the alias list.

The token definition for objectClassT shows the inheritance hierarchy for the nisMailAlias object class.

In the Build section, the DN of the entry is created by concatenating the cn attribute and attribute value with the `BASE_DN` token. The `BASE_DN` token identifies the naming context under which the entry is created.

The rfc822mailMember and objectClass attributes are multi-valued. There is one occurrence of these attributes for each value resulting from the `string2instances` operations.

For example, the `aliases` file in the domain France.XYZ.com contains the following line:

```
dir-team: pdurand, jdupond, asantini, msmith, dphilippe, smartin
```

The DN of the directory entry created from this line in the `aliases` file is cn=dir-team, ou=Aliases, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	dir-team
rfc822mailMember	pdurand
	jdupond
	asantini
	msmith
	dphilippe
	smartin
objectClass	top
	nisMailAlias

bootparams

An entry in the `bootparams` file is usually of the form:

```
hostname  parameter1 parameter2 parameter3...parameterx
```

For example, the `bootparams` file in the domain France.XYZ.com contains the following line:

```
camembert root=server1:/export/camembert/root \
  swap=server1:/export/camembert/swap \
 domain=France.XYZ.com
```

The DN of the directory entry created from this line in the `bootparams` file is cn=camembert, ou=Hosts, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	camembert
bootParameter	root=server1:/export/roquefort/root
	swap=server1:/export/roquefort/swap
	domain=France.XYZ.com
objectClass	top
	device
	bootableDevice

The host camembert probably also has an entry in `/etc/ethers` and in `/etc/hosts`. However, in the LDAP directory, the host camembert has just one entry, with all the attributes derived from the ethers table mapping and the hosts table mapping. The LDAP entry created for camembert has several object classes: one inherited structural object class, device, and three auxiliary object classes, bootableDevice, ieee802Device, and ipHost.

Based on the examples given in “ethers” on page 99, and in “hosts” on page 100, the complete entry created in the LDAP directory for host camembert is:

cn	camembert
	bertie
bootParameter	root=server1:/export/roquefort/root
	swap=server1:/export/roquefort/swap
	domain=France.XYZ.com
macAddress	0:1:23:aa:bb:cc
ipHostNumber	123.456.789.1
description	SS5 Pierre’s Desktop

objectClass	top
	device
	bootableDevice
	ieee802Device
	ipHost

If the entry for camembert is deleted from the `bootparams` file, the directory entry for camembert is updated by removing the bootableDevice object class and the bootParameter attributes which are specific to that object class.

ethers

An entry in the `ethers` file is usually of the form:

```
ethernetaddress machinename
```

For example, the `ethers` file in the domain France.XYZ.com contains the following line:

```
0:1:23:aa:bb:cc camembert
```

The DN of the directory entry created from this line in the ethers file is cn=camembert, ou=Hosts, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	camembert
macAddress	0:1:23:aa:bb:cc
objectClass	top
	device
	ieee802Device

The host camembert may also have an entry in the `/etc/bootparams` file and the `/etc/hosts` file. See the example given in "bootparams" on page 97 for details of the complete LDAP directory entry created from these files.

group

An entry in the `group file is usually of the form:`

```
groupname:password:groupidnumber:listofmembers
```

For example, the `group` file in the domain France.XYZ.com contains the following line:

```
sysadmin:yai957KJwXrjc:10:bgreen, hgrant, dbrown
```

The DN of the directory entry created from this line in the `hosts` file is cn=sysadmin, ou=Group, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	sysadmin
gidNumber	10
memberUid	bgreen
	hgrant
	dbrown
userPassword	{crypt}yai957KJwXrjc
objectClass	top
	posixGroup

hosts

An entry in the `hosts` file is usually of the form:

```
ipaddress hostname hostaliasnames #hostdescription
```

For example, the `hosts` file in the domain France.XYZ.com contains the following line:

```
123.456.789.1 camembert bertie # SS5 Pierre’s Desktop
```

The DN of the directory entry created from this line in the `hosts` file is cn=camembert, ou=Hosts, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	camembert
	bertie
ipHostNumber	123.456.789.1
description	SS5 Pierre’s Desktop

objectClass	top
	device
	ipHost

The host camembert may also have an entry in the `/etc/bootparams` file and the `/etc/ethers` file. See the example given in “bootparams” on page 97 for details of the complete LDAP directory entry created from these files.

netgroup

An entry in the `netgroup` file is usually of the form:

```
netgroupname grouptriple grouptriple grouptriple grouptriple
```

where `grouptriple` is of the form (`hostname`, `username`, `domainname`).

For example, the `netgroup` file in the domain France.XYZ.com contains the following line:

```
printers\
 (bordeaux,-,France.XYZ.com)\
 (bourgogne,-,France.XYZ.com)\
 (sauternes,-,France.XYZ.com)
```

The DN of the directory entry created from this line in the `netgroup` file is cn=printers, ou=netgroup, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	printers
nisNetGroupTriple	bordeaux,-,France.XYZ.com
	bourgogne,-,France.XYZ.com
	sauternes,-,France.XYZ.com
objectClass	top
	nisNetGroup

networks

An entry in the `networks` file is usually of the form:

```
networkname networkaddress networkalias #description
```

For example, the `networks` file in the domain France.XYZ.com contains the following line:

```
XYZ-eng 123.456.789 eng #engineering subnetwork
```

The DN of the directory entry created from this line in the `networks` file is cn=XYZ-eng, ou=networks, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	XYZ-eng
	eng
ipNetworkNumber	123.456.789
description	engineering subnetwork
objectClass	top
	ipNetwork

passwd

An entry in the `passwd` file is usually of the form:

```
userid:userPasswd:uidnumber:gidnumber:gecos:homeDir:shell
```

When there is a `shadow` file associated with the `passwd` file, it is usually of the form:

```
userid:userPasswd::::::
```

The user ID in the `passwd` file and in the `shadow` file are identical, but the user's password is actually stored in the `shadow` file and not in the `passwd` file.

For example, the `passwd` file in the domain France.XYZ.com contains the following line for Pierre Durand:

```
pdurand:x:12345:67:Pierre Durand - Project Manager:/home/pdurand:/bin/csh
```

The `x` instead of the user password indicates that the actual password is stored in the `shadow` file. The `shadow` file contains the following line for Pierre Durand:

```
pdurand:yai957KJwXrjc:::::::
```

The DN of the directory entry created from these lines in the `passwd` file and in the `shadow` file is uid=pdurand, ou=People, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	Pierre Durand
uid	pdurand
userPassword	{crypt}yai957KJwXrjc
uidNumber	12345
gidNumber	67
gecos	Pierre Durand - Project Manager
homeDirectory	/home/pdurand
loginshell	/bin/csh
objectClass	top
	account
	posixAccount

protocols

An entry in the `protocols` file is usually of the form:

```
protocolname protocolnumber  protocolalias  #description
```

For example, the `protocols` file in the domain France.XYZ.com contains the following line:

```
tcp 0 TCP_Protocol  #Transmission Control Protocol
```

The DN of the directory entry created from this line in the `protocols` file is cn=tcp, ou=protocols, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	tcp
	TCP_Protocol
ipProtocolNumber	0
description	Transmission Control Protocol
objectClass	top
	ipProtocol

rpc

An entry in the `rpc` file is usually of the form:

```
programname programnumber protocolalias #description
```

For example, the `rpc` file in the domain France.XYZ.com contains the following line:

```
yppasswdd 100123 yppasswd
```

The DN of the directory entry created from this line in the `rpc` file is cn=yppasswdd, ou=rpc, ou=Services, dc=France, dc=XYZ, dc=com. The attributes stored under that entry and their values are:

cn	yppasswdd
	yppasswd
oncRpcNumber	100123
objectClass	top
	oncRpc

ypservers

In the NIS environment, the `ypservers` file contains a list of the NIS servers in the domain.

For example, the `ypservers` file in the domain France.XYZ.com contains the following list:

```
brie
camembert
emmental
gorgonzola
roquefort
```

A directory entry is created for each server listed in the file. For example, the DN of the entry created for the server brie is cn=brie, ou=ypservers, ou=Services, dc=France, dc=XYZ, dc=com. For example, the entry for the server brie contains:

cn	brie
objectClass	top
	sunNisServer

ACLs on NIS Information

NIS information in the LDAP directory is protected by a special set of ACLs. These are part of the `dsserv.acl.conf` file. The extract from this file is shown below.

```
# NIS ACLs
access to attrs=userPassword
        by self write
        by * compare

access to attrs=gecos,loginShell
        by self write
```

By default, users have read permission on all attributes in their own entry, although they have write permission only on the userPassword, gecos, and loginShell attributes.

Configuring Naming Contexts

The naming contexts created for NIS entries during the initialization of the NIS service are specified in the `nis.mapping` file by the keyword `BASE_DN`. This base DN is the concatenation of an organizational unit (ou) specific to each map, and of a rootTree token that is usually common to several maps.

For example, the naming context for the entries created from the networks maps is defined by the following two lines in the `nis.mapping` file:

```
rootTreeT=ou=Services,$NAMING_SUFFIX||ou=Services,$DC_NAMING
BASE_DN=ou=Networks,$rootTreeT
```

The directory entries created from the networks file are created under the ou=Networks, ou=Services naming context.

The choice of a naming structure through the `NAMING_SUFFIX` keyword or `DC_NAMING` keyword is a configuration decision. The `DC_NAMING` keyword contains a domain component (DC) suffix. The DNs of entries created with that naming structure have a suffix of the form dc=XYZ, dc=com. This is the default choice when you initialize the NIS service, because the import process derives A DC naming suffix from the domain name you supply during the `dsypinstall` process.

If you prefer to use a different suffix, you must un-comment the `NAMING_SUFFIX` keyword at the beginning of the `nis.mapping` file, under the Common section for the front-end. Change the value of the `NAMING_SUFFIX` keyword to specify the suffix, or naming context under which you want NIS entries to be created. The value you specify must be a valid naming context in the data store. Refer to “Creating or Modifying a Data Store” on page 43 for information on creating a naming context.

Do not comment out the `DOMAIN_NAME` keyword in the `nis.mapping` file. This keyword contains the domain name that you supplied during the `dsypinstall` process. It is used by the server to provide the NIS service.

Case-Sensitivity

As a general rule, the directory server performs a case-sensitive search if the attribute on which the search is performed is case sensitive (ces syntax). Similarly, a search is not case sensitive if the attribute on which the search is performed is case insensitive (cis syntax).

However, it is possible to force case-sensitivity over attribute syntax by using the keyword `CASE_SENSITIVE`=yes in the mapping definition of a table in the `nis.mapping` file.

For example, the uid attribute has a cis syntax. In this case, for security reasons, it is important to force case sensitivity. Therefore, the `passwd.byname` definition in the `nis.mapping` file, includes the `CASE_SENSITIVE` keyword:

```
Table: passwd.byname

 Common:
  MAP_NAME=passwd.byname
  # The LDAP attribute used to store the key (uid) is case insensitive!
  CASE_SENSITIVE=yes
```

CHAPTER **7**

Using the RADIUS Server

The RADIUS server provided with Sun Directory Services offers an authentication service for remote users. It also collects accounting information on remote user connections.

This chapter presents the RADIUS authentication and RADIUS accounting architecture, and explains how to configure the RADIUS server. It also describes the mapping of RADIUS attributes to LDAP directory attributes.

RADIUS Authentication Architecture

The RADIUS server provided with the Sun Directory Services is an authentication and authorization information server for a Network Access Server (NAS). A NAS is a device that provides an access point to the network for remote users connecting using SLIP, PPP, or any other remote access protocol. The NAS transmits the information provided in the connection request from the remote user to the RADIUS server. The RADIUS server checks this information against the entry for the remote user in the directory. It then returns to the NAS an authorization or denial for the remote user connection. It can also provide the appropriate connection parameters for the remote user connection.

The RADIUS server logs information on remote user access requests in the `dsradius.log` file in the directory `/var/opt/SUNWconn/ldap/log.`

Note - A NAS is also often referred to as a remote access server or RAS.

The authentication architecture is illustrated in Figure 7–1.

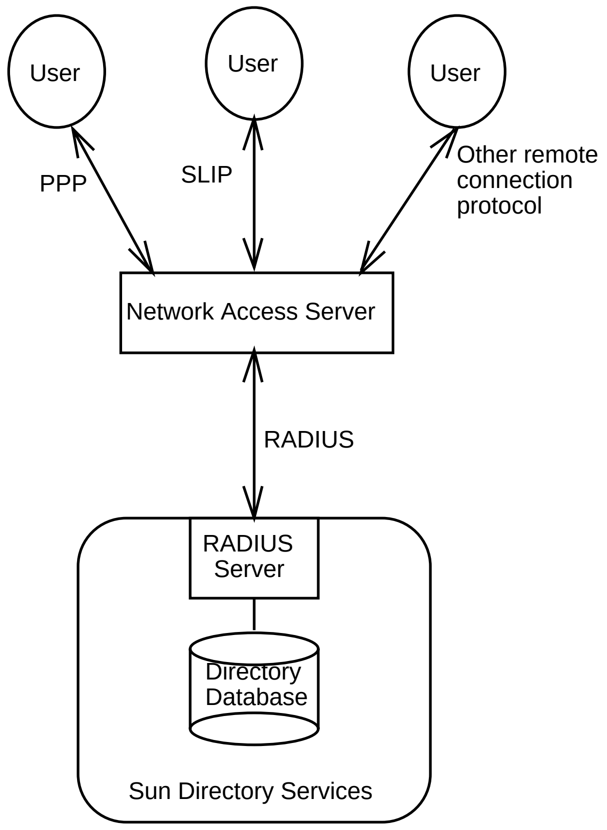


Figure 7–1 RADIUS Authentication Architecture

The *user* is the entity requesting access to network resources. In the directory database, a user is identified by a unique *uid*. The uid attribute, and all other attributes describing a remote user, are defined in the remoteUser object class.

The Network Access Server, also called a *client*, is the device to which remote users connect. The client queries the RADIUS server for authentication status, user profiles and authorizations. In the directory database, a client is identified by a unique *ipHostNumber*. The ipHostNumber attribute, and all other attributes describing a RADIUS client are defined in the nas object class.

The RADIUS server authenticates the NAS, then checks the remote user's identity and authorization in the directory database. It returns the user's status and configuration information to the NAS.

If the RADIUS server is unable to authenticate the NAS, the request from the NAS is ignored. The RADIUS server does not respond, even with a connection rejection.

RADIUS Accounting

A NAS can send accounting information on remote user connections to the RADIUS server. This information is logged separately for each NAS in a log file called `detail` in a log directory called `/var/opt/SUNWconn/ldap/radacct/`*nasname*, where *nasname* is the value of the common name (cn) attribute in the directory entry for the NAS.

If the RADIUS server is unable to authenticate the NAS, accounting information is nonetheless logged, although it is marked as `unverified` in the *nasname*`/detail` file.

All the accounting information provided by the NAS is logged.

Accounting information can also be logged dynamically in the remote user's directory entry: it is added when the user connects, and deleted when the user disconnects.

To configure the RADIUS server to log dynamic accounting information, refer to "Configuring the RADIUS Server" on page 111.

With dynamic accounting enabled, the following attributes are automatically added to a remote user's entry when the user connects, and removed when the user disconnects:

- Dynamic IP address — the IP address assigned to the remote user connection
- Dynamic session ID —the accounting session ID assigned to a remote user for a given session
- Dynamic session counter —the number of concurrent open sessions
- Dynamic IP address binding — the association between the IP address and the accounting session ID for a given session

You can specify other attributes to be added dynamically to remote users' entries by listing them in the `acctattr` file. Refer to "Configuring Dynamic Accounting" on page 117 for details of how to modify this file.

RADIUS Entries in the Directory

In the directory, there are two types of RADIUS entries, represented by two object classes:

- nas object class for NAS entries
- remoteUser object class for remote user entries

The nas object class inherits from the device object class. The mandatory attributes of the nas object class are iphostNumber and sharedKey. Refer to “Attribute Reference” on page 148 for a description of these attributes.

The optional attributes of the nas object are dictionaryFile and acctattrFile. The use of these attributes is described in detail in “Specifying a Dictionary File” on page 116, and “Configuring Dynamic Accounting” on page 117.

The remoteUser object class is an auxiliary object class that can be used with any structural object class, for example the person or organizational person object class. The remoteUser object class contains just two mandatory attributes, uid and groupCheckInfo. The uid is always passed in the connection request transmitted by the NAS to the RADIUS server. It is the key attribute used in the search filter applied by the RADIUS server to look for the remote user’s entry in the directory. The groupCheckInfo attribute lists the attributes (except uid) that the RADIUS server must check before granting access to a user.

The optional attributes of the remoteUser object class are the LDAP translation of the RADIUS attributes. They define all the possible connection parameters that can be passed in a connection request transmitted by the NAS to the RADIUS server. The RADIUS server will check these parameters against the values of the attributes stored in the directory entry for the remote user, provided they are listed in the groupCheckInfo attribute.

For the full list of optional attributes of the remoteUser object class and their description, refer to “Object Class Reference” on page 131 and “Attribute Reference” on page 148.

Initializing the RADIUS Server

When you initialize the RADIUS server, both authentication and accounting are enabled. There are two separate `dsradiusd` processes for authentication and accounting.

You can initialize the RADIUS server provided by Sun Directory Services in two ways:

- From the Admin Console
- From the command line

To initialize the RADIUS server from the Admin Console:

♦ **In the Status section, highlight RADIUS in the list of services, and click Start.**

The RADIUS service is started.

To initialize the RADIUS server from the command line:

♦ **As `root`, run this script to start the `dsradiusd` daemon:**

```
# /opt/SUNWconn/ldap/sbin/dsradius start
```

Configuring the RADIUS Server

Through the Admin Console, you can configure the following parameters for the RADIUS service:

- The naming contexts that the RADIUS server will search for user information and for NAS information. These naming contexts must be part of the tree managed by the directory server.
- Blocking Mode: that is the number of permitted consecutive failures to authenticate a user based on the password provided. Any further attempt is systematically blocked, even if the connection parameters supplied are correct. The count is reset on first success. To disable blocking mode, write zero in the Blocking Mode field.
- Whether you want to record dynamic accounting data in the directory entries of remote users (see "Configuring Dynamic Accounting" on page 117).
- The maximum number of threads that the RADIUS server will create.
- The ports on which the RADIUS processes listen for authentication requests and accounting reports.
- The directory where accounting information is stored.
- The maximum time a client will wait for a response from the RADIUS server.

The directory server queried by the RADIUS server is always localhost.

To configure these parameters, go to the RADIUS section in the Admin Console. The configuration is saved in the mapping file `/etc/opt/SUNWconn/ldap/current/mapping/radius.mapping`.

Note - Blocking mode and dynamic accounting settings are not taken into account when the RADIUS search is performed on a referral server.

Advanced Configuration of RADIUS Searches

Through the Admin Console, you can perform a basic configuration of the naming contexts that the RADIUS server will search for user and NAS information. You can manually edit the `radius.mapping` file to:

- Provide temporary access to a remote user whose entry is not in the naming context normally searched for users
- Associate remote users with a particular NAS, and grant access through that NAS only
- Combine the options above to provide temporary access through a specified NAS
- Manage remote users connecting from a virtual domain

The naming contexts searched by the RADIUS server are specified in the `/etc/opt/SUNWconn/ldap/current/mapping/radius.mapping` file. In the Dynamic section of the USERS table, the following lines define access for remote users:

```
BaseDN= l=Paris, o=xyz, c=us
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
```

In this example, the RADIUS server searches for the user ID passed in the request from the NAS in the naming context l=Paris, o=XYZ, c=US among all entries with an object class of remoteUser.

Providing Temporary Access

Without changing your basic configuration, you can allow temporary access to a remote user whose entry is in a different naming context.

1. **In the Dynamic section of the USERS table, add a BaseDN and FILTER token to the configuration with the prefix TMP_ , and assign temporary values, using the following format:**

```
TMP_BASEDN = new_search_base
   TMP_Filter = (&(Objectclass=remoteUser)(uid=$UserID)(uid=userid))
```

   where:

   - *new_search_base* is the naming context that holds the remoteUser entry for the person to whom you are granting temporary access. If this naming context is stored on a different server, ensure that a referral is defined between the two servers
   - *userid* is the actual userid of the remote user. This ensures that you grant access to that user alone, and not to all the entries with the object class remoteUser in the new search base

2. **Restart the** `dsradiusd` **daemon so that the new configuration file is taken into account.**

```
# /opt/SUNWconn/ldap/sbin/dsradius stop
   # /opt/SUNWconn/ldap/sbin/dsradius start
```

For example, if your Base DN for remote users is l=Paris, o=XYZ, c=US, and you want to provide temporary access to the remote user Felipe Gonzalez located in Madrid, you would change the local `radius.mapping` file to include:

```
BaseDN= l=Paris, o=xyz, c=us
   TMP_BaseDN= l=Madrid, o=xyz, c=us
   FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
   TMP_Filter=(&(Objectclass=remoteUser)(uid=$UserID)(uid=fgonzalez))
```

This example assumes that a referral exists between the local directory server and the directory server holding the naming context l=Madrid, o=XYZ, c=US.

If you want to temporarily extend permission to all remote users within the XYZ corporation, you would use the following temporary base DN token:

```
TMP_BaseDN= o=xyz, c=us
```

This example assumes that there is a default referral from the server that holds the l=Paris, o=XYZ, c=US data store to the server that holds the o=XYZ, c=US data store. It also assumes that the o=XYZ, c=US data store contains referral entries to all data stores held on other servers.

Restricting Access through a Specified NAS

You may want to ensure that remote users always connect to a specific NAS. For example, if you want to control communications costs, you can ensure they connect to the NAS that is geographically closest to them.

1. **In the Dynamic section of the USERS table, add a BaseDN and FILTER token to the configuration with the suffix _*nasname*, and assign temporary values, using the following format:**

```
BASEDN_nasname= search_base
   Filter_ nasname= (&(Objectclass=remoteUser)(uid=$UserID))
```

   where:

   - *nasname* is the name of the NAS (value of the cn attribute in the directory entry for the NAS)
   - *search_base* is the naming context that holds the directory entries for the remote users to whom you are granting access through the NAS

2. **Restart the `dsradiusd` daemon so that the new configuration file is taken into account.**

```
# /opt/SUNWconn/ldap/sbin/dsradius stop
   # /opt/SUNWconn/ldap/sbin/dsradius start
```

For example, your Base DN for remote users is l=France, o=XYZ, c=US, and you have remote users located in Paris, Lyon, and Toulouse who can connect to the network through a local NAS at each site. The NAS names are ParisNAS, LyonNAS, and ToulouseNAS respectively. You want to ensure that remote users always connect through the nearest NAS to save on communication costs.

You would change the `radius.mapping` file to include:

```
BaseDN= l=France, o=xyz, c=us
   BaseDN_ParisNAS= l=Paris, l=France, o=xyz, c=us
   BaseDN_LyonNAS= l=Lyon, l=France, o=xyz, c=us
   BaseDN_ToulouseNAS= l=Toulouse, l=France, o=xyz, c=us
   FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
```

When the RADIUS server receives a request from ParisNAS, it checks that the remote user belongs to the naming context l=Paris, l=France, o=XYZ, c=US.

Combining Temporary Access and NAS Restriction

You can combine temporary access permission and restrict access to a particular NAS by combining the TMP_ prefix and *_nasname* suffix on the BaseDN or the FILTER tokens.

For example, if you want to grant Felipe Gonzalez from Madrid remote access to the Paris NAS just for the duration of a business trip to Paris, you would modify the `radius.mapping` file to include the following lines in the Dynamic section of the USERS table:

```
BaseDN= l=France, o=xyz, c=us
TMP_BaseDN= l=Madrid, o=xyz, c=us
FILTER=(&(Objectclass=remoteUser)(uid=$UserID))
TMP_Filter_ParisNAS=(&(Objectclass=remoteUser)(uid=$UserID)(uid=fgonzalez))
```

In this example, the *_nasname* suffix is added to the temporary filter rather than to the temporary base DN. The reason is that you may want to grant other people from the Madrid office access through a different NAS from the Paris NAS. In this case, the temporary base DN remains valid, you just need to create the temporary filter with the appropriate *_nasname* suffix.

Managing Virtual Domains

You can manage remote user connections from users who belong to a virtual domain, that is, a domain that you manage on behalf of another organization.

For example, if ABC corporation decided to use ISP corporation to manage their internet mail service, ABC would be assigned a domain name such as abc.com, and a pool of IP addresses. ISP corporation manages user information, and remote user connections for ABC corporation. When an employee from ABC corporation connects to request remote access, the connection parameters are the user login and the user password.

For example, John Smith logs in with the following parameters:

- Login: jsmith@abc.com
- Password: secret

The RADIUS server of ISP corporation needs to separate the user ID from the domain information. The beginning of the USERS table and the variables defined in the Dynamic section are shown below:

```
Table: USERS
 Common:
  BaseDN= o=isp, c=us
 Dynamic
  userID=>$myID@$virtualDomainT || $myID
  FILTER=(&(Objectclass=remoteUser)(uid=$myID))
```

In this configuration example, the username variable accepts two alternative expressions so that it can handle equally well remote users who have a domain name appended to their user ID, and those who do not.

The domain name must be checked during the authentication procedure, therefore the directory entry of John Smith includes these attributes:

- uid: jsmith
- userPassword: * (protected)
- authSuffixName: @abc.com
- grpCheckInfo: userPassword, authSuffixName

Processing Order for RADIUS Search Parameters

During a search, the RADIUS server handles the BaseDN and FILTER tokens in the following manner: it first performs an ordinary search, then, if this search fails, it performs a search on temporary tokens.

The ordinary search starts from the most restrictive to the most general:

- FILTER_*nasname*
- FILTER
- BaseDN_*nasname*
- BaseDN

If the ordinary search fails, the temporary search is handled in the same way:

- TMP_FILTER_*nasname*
- TMP_FILTER
- TMP_BaseDN_*nasname*
- TMP_BaseDN

Specifying a Dictionary File

The RADIUS server uses a dictionary file to analyze the parameters passed in the request from the NAS. The dictionary file contains RADIUS attribute and value pairs. A number of these attributes are defined in *RFC 2138 Remote Authentication Dial In User Service*, and *RFC 2139 RADIUS Accounting*. However, NAS vendors have also defined proprietary attributes.

Do not confuse the RADIUS dictionary file with the RADIUS mapping file which provides a translation between RADIUS attributes and LDAP attributes. For information on the RADIUS mapping file, refer to "RADIUS-to-LDAP Mapping" on page 119.

Sun Directory Services is provided with a default dictionary that contains the standard attribute and value definitions. It also accepts the dictionaries from the following vendors:

- Livingston
- Ascend
- Cisco
- Shiva
- Bay Networks

The dictionary files provided by vendors contain both standard and proprietary definitions. Attribute and value definitions are identified by an OID which is the actual information passed in a RADIUS transaction. Due to a lack of standardization some proprietary attributes defined by different vendors use the same OID.

The RADIUS server can support any number of dictionary files from different vendors, but you must specify which dictionary to use with a particular NAS.

To Specify a Dictionary File for a NAS

♦ **Use the Deja tool to add the dictionaryFile attribute to the directory entry for the NAS.**

The value you assign to this attribute must be the filename of the dictionary that the RADIUS server must use for communications with the NAS described by the entry.

If the dictionaryFile attribute is not specified, the default `dictionary` file is used. This file is located with all other configuration files under `/etc/opt/SUNWconn/ldap/current`.

Note - If you use the dictionary provided by the NAS vendor instead of the default dictionary provided with Sun Directory Services, you must copy the attributes used internally by the RADIUS server from the default dictionary to the vendor-supplied dictionary. The list of attributes that you must copy is shown in Table 7–1.

TABLE 7–1 RADIUS Server Internal Attributes

```
# Non-Protocol Attributes
# These attributes are used internally by the server
#
ATTRIBUTE Expiration 21 date
ATTRIBUTE Auth-Type 1000 integer
ATTRIBUTE Menu 1001 string
ATTRIBUTE Termination-Menu 1002 string
ATTRIBUTE Prefix 1003 string
ATTRIBUTE Suffix 1004 string
ATTRIBUTE Group 1005 string
ATTRIBUTE Crypt-Password 1006 string
ATTRIBUTE Connect-Rate 1007 integer
#
#     New SUN-DS Attributes for LDAP Integration
#
ATTRIBUTE Login-Profile  2000  integer
ATTRIBUTE Login-Passwd  2001  string
ATTRIBUTE Login-Expiration  2002  date
ATTRIBUTE PPP-Profile  2010  integer
ATTRIBUTE PPP-Passwd  2011  string
ATTRIBUTE PPP-Expiration  2012  date
ATTRIBUTE SLIP-Profile  2020  integer
ATTRIBUTE SLIP-Passwd  2021  string
ATTRIBUTE SLIP-Expiration  2022  date
ATTRIBUTE Auth-Failed-Access  2100  integer
ATTRIBUTE Dynamic-Address-Profile 2200 integer
ATTRIBUTE Dynamic-Session-Counter 2201 integer
ATTRIBUTE Dynamic-SessionId 2202 string
ATTRIBUTE Dynamic-IPAddress 2203 ipaddr
ATTRIBUTE Dynamic-IPAddr-Binding 2204 string
```

Configuring Dynamic Accounting

You can use the RADIUS server to record connection parameters dynamically in the directory entry of a remote user. To enable dynamic accounting, in the RADIUS section of the Admin Console, set the Dynamic Data option to On.

The parameters recorded are dynamicIPaddress, dynamicSessionId, dynamicSessionCounter, dynamicIPaddressBinding, and all RADIUS attributes listed in the `acctattr` file. You must make sure that the NAS can provide the accounting parameters listed in the `acctattr` file. This file is located with other configuration files in `/etc/opt/SUNWconn/ldap/current`.

The dynamic accounting parameters listed in the `acctattr` file are RADIUS attributes that could be part of RADIUS accounting packets. The corresponding LDAP attributes are shown in Table 7–2. The default `acctattr` file contains examples of suitable RADIUS attributes commented out, except for User-Name. These are:

- Framed-IP-Address
- User-Name
- Acct-Session-Id
- NAS-Port
- NAS-Port-Type
- NAS-IP-Address

If you want to add accounting items to the list, check the following:

- At least one NAS can provide these items in an accounting packet
- There is an LDAP attribute for each RADIUS parameter that you want to record. If there is not, you must create the corresponding LDAP attribute. You must also create an entry in the mapping file to define the mapping between the RADIUS attribute and the LDAP attribute.

To Create a Dynamic Accounting Attribute

1. **Create an LDAP attribute for the connection parameter that you want to record.**

   This modifies the schema. See "To Create a New Attribute" on page 130.

2. **Add the attribute to the list in the** `radius.mapping` **file using a text editor.**

   Make sure you add it in both the Import section and the Export section of the file. You need to be logged in as `root` to perform this operation.

3. **Add the attribute to the list in the** `acctattr` **file using a text editor.**

   You need to be logged in as `root`.

4. **Restart the** `dsservd` **daemon so that the new accounting attribute is logged in the log file and recorded dynamically in remote user entries.**

5. **Restart the** `dsradiusd` **daemon so that the new mapping file is taken into account.**

To Specify an `acctattr` File for a NAS

♦ **Use the Deja tool to add the acctattrFile attribute to the directory entry for the NAS.**

The value you assign to this attribute must be the filename of the dynamic accounting attribute file that the RADIUS server must use to interpret the dynamic accounting information received from the NAS described by the entry.

If the acctattrFile attribute is not specified, the default `acctattr` file is used. This file is located with all other configuration files under `/etc/opt/SUNWconn/ldap/current`.

ACLs on RADIUS Information

RADIUS information in the LDAP directory is protected by a special set of ACLs. These are part of the `dsserv.acl.conf` file. The extract from this file is shown below.

```
# Radius ACLs
access to
attrs=chapPassword,radiusLoginPasswd,radiusPppPasswD,radiusSlipP
asswd
 by self write
 by * compare

access to attrs=sharedKey
 by self write
 by * compare
```

By default, users have write permission on the password and security attributes in their own entry, and read permission on all other attributes. All other users have compare permission on password and security attributes, and read permission on all other attributes.

RADIUS-to-LDAP Mapping

The RADIUS attributes and values defined in all of the dictionary files provided with the Sun Directory Services are mapped onto LDAP attributes. This mapping is defined in `/etc/opt/SUNWconn/ldap/current/mapping/radius.mapping`.

There is a one-to-one correlation between RADIUS attributes and LDAP attributes, therefore, the mapping syntax is very simple. You can easily add proprietary RADIUS attributes to the default mapping provided with Sun Directory Services.

The `radius.mapping` file is used by the RADIUS server to perform searches in the LDAP directory.

Default Mapping

Table 7–2 shows the one-to-one correspondence between RADIUS attributes and LDAP attributes. The table also indicates the origin of each RADIUS attribute. There are several kinds of RADIUS attributes:

- Standard attributes
- Vendor-specific attributes
- Sun Directory Services attributes

Standard RADIUS attributes are specified in *RFC 2138 Remote Authentication Dial In User Service* and *RFC 2138 Remote Authentication Dial In User Service*. Vendor-specific attributes are defined by NAS vendors and supplied in the dictionary file they provide with the equipment.

The LDAP attributes for the RADIUS service are specified in the schema attribute file `dsserv.at.conf`, under the section heading “Sun RADIUS Attributes”. They are also listed in the schema object class file `dsserv.oc.conf`, under the comment line “Object classes for RADIUS”.

The Sun Directory Services attributes are described in “Attribute Reference” on page 148. They represent RADIUS user profiles, and dynamic accounting parameters.

TABLE 7–2 RADIUS-to-LDAP Attribute Mapping

RADIUS attribute	Origin	LDAP attribute
User-Name	RFC 2138	uid
Crypt-Password	Sun Directory Services	userPassword
CHAP-Password	RFC 2138	chapPassword
NAS-IP-Address	RFC 2138	ipHostNumber
NAS-Identifier	RFC 2138	authNASidentifier
NAS-Port	RFC 2138	authHostPortNumber
Service-Type	RFC 2138	authServiceProtocol
Framed-Protocol	RFC 2138	framedProtocol
Framed-IP-Address	RFC 2138	framedIPAddress

TABLE 7–2 RADIUS-to-LDAP Attribute Mapping *(continued)*

RADIUS attribute	Origin	LDAP attribute
Framed-IP-Netmask	RFC 2138	ipNetmaskNumber
Framed-Routing	RFC 2138	framedRouting
Filter-Id	RFC 2138	authFilterId
Framed-MTU	RFC 2138	framedMTU
Framed-Compression	RFC 2138	framedCompression
Login-IP-Host	RFC 2138	ipLoginHost
Login-Service	RFC 2138	authLoginService
Login-TCP-Port	RFC 2138	ipLoginPort
Reply-Message	RFC 2138	authReplyMessage
Callback-Number	RFC 2138	userCallbackNumber
Callback-Id	RFC 2138	userCallbackId
Framed-Route	RFC 2138	framedRoute
Framed-IPX-Network	RFC 2138	ipxNetworkNumber
State	RFC 2138	authState
Session-Timeout	RFC 2138	sessionTimeoutNumber
Idle-Timeout	RFC 2138	idleTimeoutNumber
Termination-Action	RFC 2138	authTerminationAction
Called-Station-Id	RFC 2138	authCalleddStationId
Calling-Station-Id	RFC 2138	authCallingStationId
NAS-Port-Type	RFC 2138	authHostPortType
Port-Limit	RFC 2138	authPortLimit
Acct-Status-Type	RFC 2139	acctStatusType
Acct-Delay-Time	RFC 2139	acctDelayTime

TABLE 7–2 RADIUS-to-LDAP Attribute Mapping *(continued)*

RADIUS attribute	Origin	LDAP attribute
Acct-Input-Octets	RFC 2139	acctInputOctet
Acct-Input-Packets	RFC 2139	acctInputPacket
Acct-Output-Octets	RFC 2139	acctOutputOctet
Acct-Output-Packets	RFC 2139	acctOutputPacket
Acct-Session-Id	RFC 2139	acctSessionId
Acct-Authentic	RFC 2139	acctAuthentic
Acct-Session-Time	RFC 2139	acctSessionTime
Acct-Terminate-Cause	RFC 2139	acctTerminateCause
Expiration	Sun Directory Services	expirationDate
Auth-Type	Sun Directory Services	Auth-Type
Menu	Sun Directory Services	authStartMenuId
Termination-Menu	Sun Directory Services	authStopMenuId
Prefix	Sun Directory Services	authPrefixName
Suffix	Sun Directory Services	authSuffixName
user-check	Sun Directory Services	grpCheckInfo
user-reply	Sun Directory Services	grpReplyInfo
Login-Profile	Sun Directory Services	radiusLoginProfile
PPP-Profile	Sun Directory Services	radiusPppProfile
SLIP-Profile	Sun Directory Services	radiusSlipProfile
Login-Passwd	Sun Directory Services	radiusLoginPasswd
PPP-Passwd	Sun Directory Services	radiusPppPasswd
SLIP-Passwd	Sun Directory Services	radiusSlipPasswd
Login-Expiration	Sun Directory Services	radiusLoginExpiration

TABLE 7–2 RADIUS-to-LDAP Attribute Mapping *(continued)*

RADIUS attribute	Origin	LDAP attribute
PPP-Expiration	Sun Directory Services	radiusPppExpiration
SLIP-Expiration	Sun Directory Services	radiusSlipExpiration
Auth-Failed-Access	Sun Directory Services	radiusAuthFailedAccess
Dynamic-Session-Counter	Sun Directory Services	dynamicSessionCounter
Dynamic-SessionId	Sun Directory Services	dynamicSessionId
Dynamic-IPAddress	Sun Directory Services	dynamicIPAddress
Dynamic-IPAddr-Binding	Sun Directory Services	DynamicIPaddrBinding
Dictionary-File	Sun Directory Services	dictionaryFile
AcctAttr-File	Sun Directory Services	acctattrFile

Extending the Default Mapping

You can change the default mapping provided in `radius.mapping` to suit your own needs. If you want to add RADIUS attributes that are part of the dictionary file for your NAS to the default mapping, you must create an LDAP attribute for each RADIUS attribute missing in the `radius.mapping` file. You must then add the RADIUS-LDAP attribute pair to the `radius.mapping` file.

Creating RADIUS-LDAP Mapping Definitions

1. **Create an LDAP attribute for the RADIUS attribute you require.**

   This modifies the schema. See "To Create a New Attribute" on page 130.

2. **Add the attribute to the list in the** `radius.mapping` **file using a text editor.**

   Make sure you add it in both the Import section and the Export section of the file. You need to be logged in as `root` to perform this operation.

3. **Restart the** `dsservd` **daemon so that the modifications to the schema are taken into account, and the** `dsradiusd` **daemon so that the new mapping file is taken into account.**

4. **Run** `dejasync`**. As** `root` **type:**

```
# /opt/SUNWconn/ldap/sbin/dejasync
```

For details on the options of the `dejasync(1m)` command, refer to the man page. You must run `dejasync` if you want to use the Deja tool to modify RADIUS entries in the directory.

CHAPTER **8**

Configuring the Directory Schema

This chapter explains how to obtain the current schema from the directory server, and how to modify it to add object class or attribute definitions.

It also contains a list of all the object classes and attributes that belong to the default schema for Sun Directory Services 3.1 with a description of their purpose and meaning.

Schema Definition

The schema is the set of rules that describe the data that can be stored in the directory. It defines the type of entries, their structure and their syntax. The schema can be modified and extended, though certain objects and attributes cannot be changed.

The schema definition is stored in two files in the configuration directory `/etc/opt/SUNWconn/ldap/current`:

- `dsserv.oc.conf` defines the object classes. These specify the types of entries permitted, their superior object class, and their mandatory and optional attributes.
- `dsserv.at.conf` contains attribute definition information:
  - An oid representing the attribute
  - The attribute syntax, specified using a keyword
  - Alternate names for some attributes
  - The keyword *naming* for attributes that are naming attributes
  - The keyword *single* for single-valued attributes

For the exact format of an LDAP attribute definition, refer to the `dsserv.at.conf(4)` man page.

Displaying the Current Schema

There are two ways of displaying the current schema:

- From the command line, using the `ldapsearch` command
- In the Admin Console

▼ Displaying the Schema with `ldapsearch`

Use the `ldapsearch` command with the following arguments to read the current schema through the directory server:

```
/opt/SUNWconn/bin/ldapsearch -h hostname -b "cn=schema" -s base 'objectclass=*'
```

where *hostname* is the name of the directory server.

Table 8–1 shows an extract of the type of information returned.

TABLE 8–1 Extract from the Directory Schema

```
objectclasses=( 2.5.6.0 NAME 'top' DESC 'Standard ObjectClass'
MUST ( objectclass ) )
objectclasses=( 2.5.6.1 NAME 'alias' DESC 'Standard ObjectClass'
SUP 'top' MUST ( objectclass $ aliasedobjectname ) MAY ( * ) )
objectclasses=( 2.5.6.2 NAME 'country' DESC 'Standard ObjectClass'
SUP 'top' MUST ( objectclass $ c ) MAY ( description $ searchguide
) )
objectclasses=( 2.5.6.3 NAME 'locality' DESC 'Standard
ObjectClass' SUP 'top' MUST ( objectclass ) MAY ( description $
l $ searchguide $ seealso $ st $ street ) )
objectclasses=( 2.5.6.4 NAME 'organization' DESC 'Standard
ObjectClass' SUP 'top' MUST ( objectclass $ o ) MAY (
businesscategory $ description $ destinationindicator $
facsimiletelephonenumber $ internationalisdnnumber $ l $
physicaldeliveryofficename $ postofficebox $ postaladdress $
postalcode $ preferreddeliverymethod $ registeredaddress $
searchguide $ seealso $ st $ street $ telephonenumber $
teletexterminalidentifier $ telexnumber $ userpassword $
x121address ) )
objectclasses=( 2.5.6.5 NAME 'organizationalUnit' DESC 'Standard
ObjectClass' SUP 'top' MUST ( objectclass $ ou ) MAY (
businesscategory $ description $ destinationindicator $
facsimiletelephonenumber $ internationalisdnnumber $ l $
physicaldeliveryofficename $ postofficebox $ postaladdress $
postalcode $ preferreddeliverymethod $ registeredaddress $
searchguide $ seealso $ st $ street $ telephonenumber $
teletexterminalidentifier $ telexnumber $ userpassword $
x121address ) )
objectclasses=( 2.5.6.6 NAME 'person' DESC 'Standard ObjectClass'
SUP 'top' MUST ( objectclass $ sn $ cn ) MAY ( description $
seealso $ telephonenumber $ userpassword ) )
```

When you use the `ldapsearch` command to display the schema, the keywords are shown in capitals. They introduce the following:

- NAME introduces the name of the object class.
- DESC introduces a description of the object class.
- SUP introduces the name of the superior object class.
- MUST introduces the list of mandatory attributes for an object class. In `dsserv.oc.conf` mandatory attributes are identified by the keyword *requires*.
- MAY introduces the list of optional attributes for an object class. In `dsserv.oc.conf` mandatory attributes are identified by the keyword *allows*.

▼ Displaying the Schema in the Admin Console

1. **In the Admin Console, go to the Schema section.**

   This section displays a list of object classes in hierarchical order, or in alphabetical order. Use the buttons below the pane to change the display mode.

   Click the folder icon for an object class to display its mandatory (M) and optional (O) attributes. With the hierarchical display, clicking on the folder icon of an object class will display any subclasses of that object class.

2. **To display a list of attributes, click the Attributes list button.**

   An attribute list window is displayed. It contains a five-column table that shows:

   - The name of the attribute
   - Alias names for this attribute (separated by commas)
   - The attribute syntax, identified by a keyword (see “Attribute Reference” on page 148)
   - A unique OID for the attribute
   - Whether the attribute is a naming attribute (that is, an attribute that can be used in the distinguished name of an entry), and whether it is single-valued

Modifying the Schema

You can modify the schema in the following ways:

- By creating new object classes or attributes

- By modifying object classes and attributes

It is safer to always create a new object class rather than modify an existing one. If you want to extend an existing object class, you can create an object class that inherits from the object class that you want to extend.

Deleting object classes or attributes is not advisable since there might be directory entries that use the existing definitions.

There is no automatic check that schema modifications do not invalidate entries. Therefore, to minimize the risk of entries becoming invalid, restrict your changes to the addition of object classes or attributes. You can, however, enable schema checking. For this, refer to "Schema Checking" on page 128.

The schema definition contains object classes that are used internally by the Sun Directory Services directory server or by the Sun Internet Mail Server (SIMS). The Admin Console does not permit you to modify these object classes. They are marked with the keyword `frozen` in the configuration files. You *must not* remove this keyword from any standard schema item.

Note - If you use the web gateway to allow users to browse the directory, all modification made to the schema must also be made to the `dswebtmpl.conf` file. See the `dswebtmpl.conf(4)` man page for details.

Schema Checking

Sun Directory Services provides a schema checking feature. When directory information is added or modified, the directory server checks that all mandatory attributes of the object class or inherited by the object class are present.

The schema checking options are:

- Off: no checking is performed
- Weak: a check is performed when entries are created or modified
- Strong: in addition to the previous level, a check is performed on search operations

Select the appropriate level of checking from the Schema check menu button in the Schema section of the Admin Console. The default level of checking is weak.

Note - Schema checking cannot be performed on the compatibility of object classes. For example, you could create an entry with the device object class and the person object class. The IETF standards do not enforce rules on object classes.

▼ To Create a New Object Class

1. **From the Admin Console main window, choose Class/Attribute from the Create menu.**

The Create Object Class window is displayed.

2. **Specify:**
   - The name of the new object class
   - The object identifier for the object class (optional)
   - The superior object class, from which this object class will inherit attributes

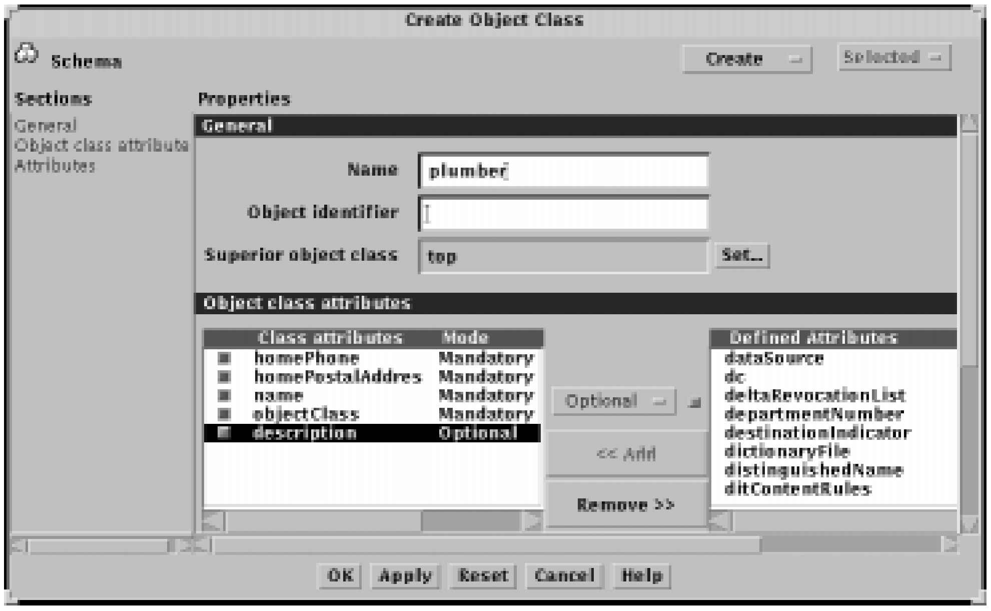
3. **Specify the mandatory and optional attributes you want to include in this class:**
   a. **Select or create the attributes you want to include in the object class.**

   b. **Select the mode of the attributes (Mandatory or Optional) from the pop-up menu.**

   c. **Click Add to add the attributes to the object class definition.**

4. **Click OK to save the modified object class definition.**

   This change will take effect when you restart the `dsservd` daemon. Figure 8–1 shows a new object class plumber, with the atttributes you would need to contact a plumber.



Name: plumber
Object identifier:
Superior object class: top

Class attributes	Mode
homePhone	Mandatory
homePostalAddres	Mandatory
name	Mandatory
objectClass	Mandatory
description	Optional

Defined Attributes
dataSource
dc
deltaRevocationList
departmentNumber
destinationIndicator
dictionaryFile
distinguishedName
ditContentRules

Figure 8–1 Create Object Class Window

▼ To Create a New Attribute

1. **From the Admin Console main window, choose Class/Attribute from the Create menu.**

   The Create Object Class window is displayed.

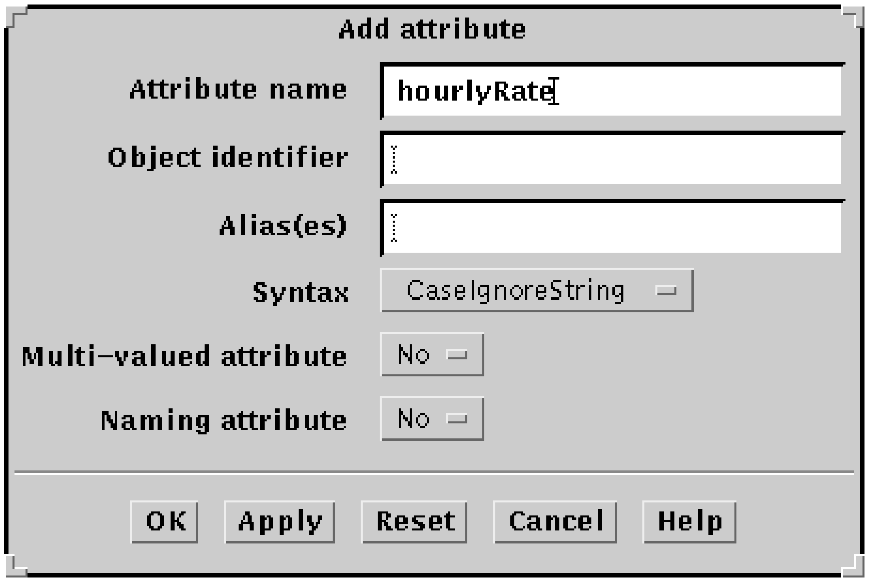
2. **In the Create Object Class window, choose Attribute from the Create menu.**

   The Add Attribute window is displayed.

3. **Specify:**
   - The name of the attribute
   - The unique OID for this attribute (optional)
   - Any alternate names in the Aliases field (optional)
   - Whether the attribute is multi-valued
   - Whether the attribute can be used as a naming attribute

4. **Click OK to save the new attribute definition.**

   This change will take effect when you restart the `dsservd` daemon. Figure 8–2 shows a new attribute hourlyRate has been created to be added to the plumber object class.



Add attribute

Attribute name	hourlyRate
Object identifier	
Alias(es)	
Syntax	CaseIgnoreString
Multi–valued attribute	No
Naming attribute	No

Figure 8–2 Create Attribute Window

▼ To Add an Attribute to an Object Class

1. **In the object class list, highlight the object class to which you want to add an attribute, and choose Modify Class/Attribute from the Selected menu.**

   The Modify Object Class window is displayed. The name of the object class you are modifying is displayed in the General section of this window. The mandatory and optional attributes for that object class are listed in the Object class attributes section.

2. **In the Defined Attributes list, highlight the attribute that you want to add.**

3. **Select the mode of the attribute (Mandatory or Optional) from the pop-up menu.**

4. **Click Add to add the attribute to the object class definition.**

5. **Click OK to save the modified object class definition.**

   This change will take effect when you restart the `dsservd` daemon.

   To change the mode of an attribute that is already included in the object class definition, select the attribute in the Class attributes list and change the mode using the Mode pop-up menu.

Object Class Reference

This section contains an alphabetical list of the object classes accepted by the default schema, except for pilot project object classes described in *RFC 1274 The COSINE and Internet X.500 Schema*. It explains the purpose of each object class, and gives the list of mandatory and optional attributes specific to the particular object class. An object class also inherits the mandatory and optional attributes from its superior object class. Inherited attributes are not listed.

The keyword *frozen* after the object class name indicates that this object class is used by a component of Sun Internet Mail Server, or by a component of Sun Directory Services. You cannot change a frozen object class definition using the Admin Console. If you change the definition of such an object class, ensure that your changes do not prevent the Sun Internet Mail Server and the Sun Directory Services components from using objects of this class.

account

- Description: Used to define entries representing a user account.
- Superior object class: top
- Mandatory attribute: uid
- Optional attributes: description, host, l, o, ou, seeAlso

alias (frozen)

- Description: An alternative name for an object located under the same data store suffix.
- Superior object class: top
- Mandatory attribute: aliasedObjectName

Note - It is preferable to avoid using the alias object class and use instead the aliasObject subclass. This is because the alias object class only allows the full DN of the aliased object as its naming attribute, and not just the RDN.

---

aliasObject

- Description: An alternative name for an object located under the same data store suffix.
- Superior object class: alias
- Optional attributes: * (allows any attribute)

Note - The attributes in the aliasObject entry must include the naming attribute of the entry. The naming attribute should be the same as for the aliased object.

---

applicationEntity

- Description: Used to define an entry representing an application entity.
- Superior object class: top
- Mandatory attributes: cn, presentationAddress
- Optional attributes: description, l, o, ou, seeAlso, supportedApplicationContext

applicationProcess

- Description: Used to define an entry representing an application process.
- Superior object class: top

- Mandatory attribute: cn
- Optional attributes: description, l, ou, seeAlso

automount

- Description: Used to define an entry representing an NIS automount record.
- Superior object class: top
- Mandatory attributes: cn, automountInformation
- Optional attribute: description

bootableDevice (auxiliary object class)

- Description: Used to define an entry representing any device that requires boot parameters. Used to import information from the `/etc/bootparams` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: device
- Optional attribute: bootFile, bootParameter

certificationAuthority (auxiliary object class)

- Description: Used to define entries representing objects that act as certification authorities. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attributes: authorityRevocationList, cACertificate, certificateRevocationList
- Optional attribute: crossCertificatePair

certificationAuthority-V2 (auxiliary object class)

- Description: Used to define entries representing objects that act as certification authorities for version 2. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: certificationAuthority
- Optional attribute: crossCertificatePair

country

- Description: Identifies country entries in the directory.

- Superior object class: top
- Mandatory attribute: c
- Optional attributes: description, searchGuide

cRLDistributionPoint

- Description: Used to define an entry that provides a service for certification authority revocation lists.
- Superior object class: top
- Mandatory attribute: cn
- Optional attributes: authorityRevocationList, certificateRevocationList, deltaRevocationList

device

- Description: Used to define an entry representing a device (for example a modem or CD-ROM drive).
- Superior object class: top
- Mandatory attribute: cn
- Optional attributes: description, l, o, ou, owner, seeAlso, serialNumber

dcObject (auxiliary object class)

- Description: Used to define an entry representing a domain component, that is a component in the dot-separated sequence that forms a domain name. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attribute: dc

dmd

- Description: Used to define an entry that represents a directory management domain (DMD), that is the authority responsible for a particular directory server.
- Superior object class: top
- Mandatory attribute: dmdName
- Optional attributes: businessCategory, description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l,

physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, searchGuide, seeAlso, st, street, telephoneNumber, userPassword, x121Address

dNSDomain

- Description: Used to define entries representing a DNS domain.
- Superior object class: domain
- Optional attribute: dNSRecord

document

- Description: Used to define an entry representing a document.
- Superior object class: pilotObject
- Mandatory attribute: documentIdentifier
- Optional attributes: abstract, cn, description, documentAuthor, documentAuthorCommonName, documentAuthorSurname, documentLocation, documentPublisher, documentStore, documentTitle, documentVersion, keywords, l, o, obsoletedByDocument, obsoletesDocument, ou, seeAlso, subject, updatedByDocument, updatesDocument

documentSeries

- Description: Used to define an entry representing a series of related documents.
- Superior object class: top
- Mandatory attributes: cn
- Optional attributes: description, l, o, ou, seeAlso, telephoneNumber

domain

- Description: Used to define an entry representing a domain.
- Superior object class: top
- Mandatory attribute: dc
- Optional attributes: associatedName, businessCategory, description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, o, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, searchGuide, seeAlso, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, userPassword, x121Address

domainRelatedObject

- Description: Used to define an entry related to a domain.
- Superior object class: top
- Mandatory attribute: associatedDomain

dSA

- Description: Used to define an entry representing a directory system agent (DSA) or any directory server.
- Superior object class: applicationEntity
- Optional attribute: knowledgeInformation

emailGroup

- Description: Used to define an entry representing an electronic mail distribution list that uses `aliases(4)` format.
- Superior object class: top
- Mandatory attributes: cn
- Optional attributes: authorizedDomain, authorizedSubmitter, dataSource, expandable, mailDeliveryFile, mailDeliveryOption, mailProgramDeliveryInfo, mailHost, ownerDeliveryFile, ownerDeliveryOption, ownerProgramDeliveryInfo, requestsToDeliveryFile, requestsToDeliveryOption, requestsToProgramDeliveryInfo, rfc822AuthorizedSubmitter, rfc822MailMember, rfc822Owner, rfc822UnauthorizedSubmitter, unauthorizedDomain, unauthorizedSubmitter

emailPerson (frozen)

- Description: Used to define an entry for a person who uses electronic mail.
- Superior object class: inetOrgPerson
- Mandatory attributes: cn, objectClass
- Optional attributes: assistant, channelName, channelType, dataSource, generationQualifier, freeFormName, homeDirectory, homeFacsimileTelephoneNumber, mail, mailAutoReplyExpirationDate, mailAutoReplyMode, mailAutoReplySubject, mailAutoReplyText, mailAutoReplyTextInternal, mailDeliveryFile, mailDeliveryOption, mailFolderMap, mailForwardingAddress, mailHost, mailMessageStore, mailProgramDeliveryInfo, mailQuota, objectStatus, preferredRfc822Recipient, reportsTo, rfc822Mailbox, userDefinedAttribute1, userDefinedAttribute2, userDefinedAttribute3, userDefinedAttribute4

friendlyCountry

- Description: Used to allow friendlier naming of country entries than with the object class country. The naming attribute of object class country, countryName, has to be a 2 letter string defined in ISO 3166.
- Superior object class: country
- Mandatory attribute: co

gatewayCCMailUser (frozen)

- Description: Used to define an entry representing a user of Lotus CC:Mail.
- Superior object class: top
- Optional attributes: cCMailAddresses, preferredCCMailOriginator, preferredCCMailRecipient

gatewayChannel (frozen)

- Description: Used to define an entry representing a legacy mail gateway channel.
- Superior object class: top
- Mandatory attributes: channelName
- Optional attributes: ackedSequenceNumber, channelType, currentSequenceNumber, maxLastModifiedTime, objectStatus, seeAlso, userPassword

gatewayDocConvPreference (frozen)

- Description: Used to store preferences for document conversion for a gateway user.
- Superior object class: top
- Optional attribute: docConvPreference

gatewayLotusNotesUser (frozen)

- Description: Used to define an entry representing a user of Lotus Notes.
- Superior object class: top
- Optional attributes: lotusNotesAddresses, preferredLotusNotesOriginator, preferredLotusNotesRecipient

gatewayMail11User (frozen)

- Description: Used to define an entry representing a user of Mail-11 (DEC).

- Superior object class: top
- Optional attributes: mail11Addresses, preferredMail11Originator, preferredMail11Recipient

gatewayMrUser (frozen)

- Description: Used to define an entry representing a user of the legacy Mail Relay (MR) mail system.
- Superior object class: top
- Optional attributes: mrAddresses, preferredMrOriginator, preferredMrRecipient

gatewayMSMailUser (frozen)

- Description: Used to define an entry representing a user of Microsoft Mail.
- Superior object class: top
- Optional attributes: mSMailAddresses, preferredMSMailOriginator, preferredMSMailRecipient

gatewayNGMUser (frozen)

- Description: Used to define an entry representing a user of the legacy Novell Groupwise Mail (NGM) mail system.
- Superior object class: top
- Optional attributes: nGMAddresses, preferredNGMOriginator, preferredNGMRecipient

gatewayNGM70User (frozen)

- Description: Used to define an entry representing a user of the legacy Novell Groupwise Mail 7.0 (NGM70) mail system.
- Superior object class: top
- Optional attributes: nGM70Addresses, preferredNGM70Originator, preferredNGM70Recipient

gatewayPROFSUser (frozen)

- Description: Used to define an entry representing a user of IBM PROFS.
- Superior object class: top

- Optional attributes: pROFSAddresses, preferredPROFSOriginator, preferredPROFSRecipient

groupOfNames

- Description: Used to define entries representing an unordered set of names of objects or other groups.
- Superior object class: top
- Mandatory attributes: cn, member
- Optional attributes: businessCategory, description, o, ou, owner, seeAlso

groupOfUniqueNames

- Description: Used to define entries representing an unordered set of names of objects or other groups. Each name in the set is unique in the directory.
- Superior object class: top
- Mandatory attributes: cn, uniqueMember
- Optional attributes: businessCategory, description, o, ou, owner, seeAlso

ieee802Device (auxiliary object class)

- Description: Used to define entries representing any device that has a MAC address. Used to import information from the `/etc/ethers` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: device
- Optional attributes: macAddress

inetOrgPerson

- Description: Used to define an entry for a person who uses the Internet and belongs to an organization.
- Superior object class: organizationalPerson
- Optional attributes: audio, businessCategory, carLicense, departmentNumber, employeeNumber, employeeType, givenName, homePhone, homePostalAddress, initials, jpegPhoto, labeledURI, mail, manager, mobile, pager, photo, preferredLanguage, roomNumber, secretary, uid, userCertificate, userSMIMECertificate, x500uniqueIdentifier

ipHost (auxiliary object class)

- Description: Used to describe a device that has an IP address. Used to import information from the `/etc/hosts` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attributes: cn, ipHostNumber
- Optional attributes: description, bootFile, bootParameter, l, macAddress, manager, serialNumber

ipProtocol

- Description: Used to define an entry that describes an IP protocol. Used to import information from the `/etc/protocols` file.
- Superior object class: top
- Mandatory attributes: cn, ipProtocolNumber
- Optional attribute: description

ipNetwork

- Description: Used to define an entry that describes an IP network. Used to import information from the `/etc/networks` file.
- Superior object class: top
- Mandatory attributes: cn, ipNetworkNumber
- Optional attributes: description, ipNetmaskNumber, l, manager

ipService

- Description: Used to define an entry that represents an IP service.
- Superior object class: top
- Mandatory attributes: cn, ipServicePort, ipServiceProtocol
- Optional attribute: description

labeledURIObject (auxiliary object class)

- Description: Used to define an entry that describes a resource on the network that is identified by a URI. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top

- Optional attribute: labeledURI

locality

- Description: Used to define entries that describe locality.
- Superior object class: top
- Optional attributes: description, locality, searchGuide, seeAlso, st, street

nas

- Description: Used to define a Network Access Server used in the context of RADIUS authentication.
- Superior object class: device
- Mandatory attributes: iphostnumber, sharedKey
- Optional attributes: dictionaryFile, acctattrFile

nisMailAlias

- Description: Used to define an entry that represents an NIS `mail.aliases` record. Used to import information from the `/etc/mail/aliases` file.
- Superior object class: top
- Mandatory attribute: cn
- Optional attribute: rfc822MailMember

nisMap

- Description: Used to define an entry that represents an NIS map.
- Superior object class: top
- Mandatory attribute: nisMapName
- Optional attribute: description

nisNetGroup

- Description: Used to define an entry that represents an NIS netgroup record. Used to import information from the `/etc/netgroup` file.
- Superior object class: top
- Mandatory attribute: cn

- Optional attributes: description, memberNisNetGroup, nisNetGroupTriple

nisNetId

- Description: Used to define an entry that represents an NIS `netid.byname` record.
- Superior object class: top
- Mandatory attribute: cn
- Optional attribute: nisNetIdGroup, nisNetIdHost, nisNetIdUser

nisObject

- Description: Used to define an entry in the directory that represents an entry in an NIS map. The NIS key is stored in the cn attribute.
- Superior object class: top
- Mandatory attribute: nisMapName
- Optional attributes: cn, description, nisMapEntry

nisSunObject

- Description: Used to define an entry in the directory that represents an entry in an NIS map. This object class is used in the generic NIS map definition in Sun Directory Services. The NIS key is stored in the sunNisKey attribute.
- Superior object class: top
- Mandatory attribute: nisMapName
- Optional attributes: cn, description, nisMapEntry, sunNisKey

oncRpc

- Description: Used to define an entry that represents an Open Network Computing (ONC) remote procedure call (RPC). Used to import information from the `/etc/rpc` file.
- Superior object class: top
- Mandatory attributes: cn, oncRpcNumber
- Optional attribute: description

organization

- Description: Used to define organization entries in the directory.

- Superior object class: top
- Mandatory attributes: o
- Optional attributes: businessCategory, description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, searchGuide, seeAlso, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, userPassword, x121Address

organizationalPerson

- Description: Used to define entries representing people employed by, or in some way associated with, an organization.
- Superior object class: person
- Optional attributes: destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, ou, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, title, x121Address

organizationalRole

- Description: Used to define entries representing a role or position within an organization. An organizationalRole is usually filled by an organizationalPerson, but it can also be filled by a non-human entity.
- Superior object class: top
- Mandatory attribute: cn
- Optional attributes: description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, ou, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, roleOccupant, seeAlso, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, x121Address

organizationalUnit

- Description: Used to define entries representing subdivisions of an organization.
- Superior object class: top
- Mandatory attributes: ou
- Optional attributes: businessCategory, description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode,

registeredAddress, searchGuide, seeAlso, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, userPassword, x121Address

person

- Description: Used to define entries representing people.
- Superior object class: top
- Mandatory attributes: cn, sn
- Optional attributes: description, seeAlso, telephoneNumber, userPassword

posixAccount (auxiliary object class)

- Description: Used to represent an account defined by POSIX attributes. Used to import information from the `/etc/passwd` file. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attributes: cn, uid, uidNumber, gidNumber, homeDirectory
- Optional attributes: description, gecos, loginShell, userPassword

posixGroup

- Description: Used to define an entry that represents a group of POSIX accounts. Used to import information from the `/etc/group` file.
- Superior object class: top
- Mandatory attributes: cn, gidNumber
- Optional attributes: description, memberUid, userPassword

referral (frozen)

- Description: Used to define an entry that points to another data store.
- Superior object class: top
- Optional attributes: ref, * (allows any attribute, in particular the same naming attribute as in the RDN of the referenced object)

remoteUser (frozen)

- Description: In the context of RADIUS authentication, used to define remote users who access the network through a Network Access Server (NAS).

- Superior object class: top
- Mandatory attribute: uid
- Optional attributes: acctAuthentic, acctDelayTime, acctInputOctet, acctOutputOctet, acctSessionId, acctSessionTime, acctStatusType, acctTerminateCause, authCalleddStationId, authCallingStationId, authFilterId, authHostPortNumber, authHostPortType, authLoginService, authPortLimit, authPrefixName, authReplyMessage, authServiceProtocol, authType, authStartMenuId, authSuffixName, authState, authStopMenuId, authTerminationAction, chapPassword, expirationDate, framedCompression, framedIPAddress, framedMTU, framedRoute, framedRouting, framedProtocol, grpCheckInfo, grpReplyInfo, idleTimeoutNumber, ipHostNumber, ipLoginHost, ipLoginPort, ipNetmaskNumber, ipxNetworkNumber, radiusLoginProfile, radiusPppProfile, radiusSlipProfile, radiusAuthFailedAccess, radiusLoginExpiration, radiusLoginPasswd, radiusPppExpiration, radiusPppPasswd, radiusSlipExpiration, radiusSlipPasswd, dynamicSessionCounter, dynamicSessionId, dynamicIPAddress, sessionTimeoutNumber, userCallbackId, userCallbackNumber, userPassword

residentialPerson

- Description: Used to define entries representing a person in the residential environment.
- Superior object class: top
- Mandatory attribute: l
- Optional attributes: businessCategory, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, x121Address

rFC822LocalPart

- Description: Used to define entries which represent the local part of RFC822 mail addresses. This treats this part of an RFC822 address as a domain.
- Superior object class: domain
- Optional attributes: cn, description, destinationIndicator, facsimileTelephoneNumber, internationaliSDNNumber, l, o, physicalDeliveryOfficeName, postOfficeBox, postalAddress, postalCode, preferredDeliveryMethod, registeredAddress, seeAlso, sn, st, street, telephoneNumber, teletexTerminalIdentifier, telexNumber, userPassword, x121Address

room

- Description: Used to define an entry representing a room.
- Superior object class: top
- Mandatory attribute: cn
- Optional attributes: description, roomNumber, seeAlso, telephoneNumber

shadowAccount (auxiliary object class)

- Description: Used to represent a user that has a shadow password. It is an *auxiliary* object class, which means that it may be used in conjunction with any object class.
- Superior object class: top
- Mandatory attribute: uid
- Optional attributes: description, shadowLastChange, shadowMax, shadowMin, shadowWarning, shadowInactive, shadowExpire, shadowFlag, userPassword

simpleSecurityObject

- Description: Used to define an entry containing a user password, for simple authentication.
- Superior object class: top
- Mandatory attribute: userPassword

strongAuthenticationUser (auxiliary object class)

- Description: Used to define an entry for an object participating in Strong Authentication. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attribute: userCertificate

subschema (auxiliary object class)

- Description: Used to define an entry that contains the rules governing the schema. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Optional attributes: attributeTypes, dITStructureRules, ditContentRules, matchingRules, matchingRuleUse, nameForms, objectClasses

sunNisMap

- Description: Used by the NIS/LDAP server to manage NIS maps. An entry is created for each map stored in the LDAP directory.
- Superior object class: top
- Mandatory attributes: sunNisDomain, sunNisMapFullName, sunNisMapState, sunNisMaster, sunNisSecurityMode
- Optional attributes: description, seeAlso, sunNisDbmCache, sunNisDnsForwarding, sunNisInputFile, sunNisOutputName, sunNisLoadMap

sunNisServer

- Description: Used to define an entry that represents an NIS ypservers record. Used to import information from the `ypservers` file.
- Superior object class: top
- Mandatory attributes: cn

top

- Description: An abstract object class, parent of all others. It ensures that every object class contains the objectClass attribute.
- Mandatory attribute: objectClass

uidObject (auxiliary object class)

- Description: Used to name an entry with a unique ID. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Mandatory attribute: uid

userSecurityInformation (auxiliary object class)

- Description: Used to store security information about a user. It is an *auxiliary* object class, which means that it should be used in conjunction with a structural object class.
- Superior object class: top
- Optional attribute: supportedAlgorithms

Attribute Reference

All attributes defined in the schema have one of the following syntaxes:

- Distinguished name (dn)
- Case-ignore string (cis) — An alphanumeric string, not case-sensitive
- Case-exact string (ces) — A case-sensitive alphanumeric string
- Telephone number (tel)
- Integer (int or long)
- Binary (bin)
- Encrypted (protected) — A value that has been encrypted using the method specified through the Admin Console. Possible values are **`sunds`**, **`crypt`**, or **`none`**.
- UTC time (utctime)

The following list of attributes in the default schema gives the attribute syntax, any alternative names, and explains how the attribute is used.

abstract

- Syntax: cis
- Description: A brief description of the document described by the entry.

acctattrFile

- Syntax: ces
- Description: Specifies the name of the dynamic accounting attributes file to be used to interpret the dynamic accounting information received from the NAS described by the entry.

acctAuthentic

- Syntax: ces
- Description: Used in RADIUS accounting requests to indicate how the user described by the entry was authenticated.

acctDelayTime

- Syntax: ces

- Description: Used in RADIUS accounting requests to indicate for how long the NAS has been trying to send an accounting report. The delay is deducted from the time of arrival of the report to determine the actual time at which the event occurred.

acctInputOctet

- Syntax: ces
- Description: Used in RADIUS accounting requests to indicate the number of octets received during the provision of service.

acctInputPacket

- Syntax: ces
- Description: Used in RADIUS accounting requests to indicate the number of packets received during the provision of service.

acctOutputOctet

- Syntax: ces
- Description: Used in RADIUS accounting requests to indicate the number of octets sent during the provision of service.

acctOutputPacket

- Syntax: ces
- Description: Used in RADIUS accounting requests to indicate the number of packets sent during the provision of service.

acctSessionId

- Syntax: ces
- Description: Used in RADIUS accounting to provide a unique accounting ID. It is used to match start and stop records for the same session.

acctSessionTime

- Syntax: ces
- Description: Used in RADIUS accounting to indicate the number of seconds during which the user described by the entry has received service.

acctStatusType

- Syntax: ces
- Description: Used in RADIUS accounting to indicate whether the current report marks the beginning of service (start) or the end (stop).

acctTerminateCause

- Syntax: ces
- Description: Used in RADIUS accounting to indicate how a session was terminated.

ackedSequenceNumber

- Syntax: ces
- Description: A sequence number used during Legacy Mail directory synchronization.

aliasedObjectName

- Alternate name: aliasedEntryName
- Syntax: dn
- Description: The DN of the entry for which the alias entry is an alias. This attribute is single-valued.

altServer

- Syntax: cis
- Description: Specifies the URLs of other servers to contact if the current server is unavailable.

assistant

- Syntax: cis
- Description: An assistant to the person described by the entry.

associatedDomain

- Syntax: cis

- Description: The domain with which the object described by this entry is associated.

associatedName

- Syntax: dn
- Description: The distinguished name of an entry associated with this entry.

attributeName

- Syntax: cis
- Description: Specifies the name of an attribute.

attributeTypes

- Syntax: cis
- Description: Specifies the attribute types allowed in the schema.

audio

- Syntax: bin
- Description: Sound information associated with the object described by the entry.

authCalledStationId

- Syntax: ces
- Description: Indicates the phone number called by the user to request access through a NAS.

authCallingStationId

- Syntax: ces
- Description: Indicates the phone number from which the used called to request access through a NAS.

authFilterId

- Syntax: ces
- Description: Indicates the name of the filter list for the user described by the entry.

authHostPortNumber

- Syntax: ces
- Description: Indicates the physical port number of the NAS that is authenticating the user.

authHostPortType

- Syntax: ces
- Description: Indicates the type of physical port number of the NAS that is authenticating the user.

authLoginService

- Syntax: ces
- Description: Indicates the service that should be used to connect the user to the login host.

authNASidentifier

- Syntax: ces
- Description: Contains a string that identifies the NAS that transmitted an access request.

authorityRevocationList

- Syntax: cis
- Description: A list of certificates that have been revoked by the certification authority described in the entry, or that the certification authority knows have been revoked by other certification authorities.

authorizedDomain

- Syntax: cis
- Description: Domain name from which users are authorized to post to the list described by the entry.

authorizedSubmitter

- Syntax: cis

- Description: A registered user authorized to post messages to the list described by the entry.

authPortLimit

- Syntax: ces
- Description: Sets the maximum number of ports to be provided by the NAS to the user.

authPrefixName

- Syntax: ces
- Description: Used internally by the RADIUS server to distinguish between the user name to be processed for authentication and a possible prefix. In some cases, the connection protocol can add a prefix to the user’s name, for example, ppp%jsmith.

authReplyMessage

- Syntax: cis
- Description: Contains text that the NAS can display to the user.

authServiceProtocol

- Syntax: ces
- Description: Indicates the type of service requested by the user.

authStartMenuId

- Syntax: ces
- Description: This attribute is used internally by the RADIUS server.

authState

- Syntax: ces
- Description: A state attribute sent by the RADIUS server to the NAS. The NAS must send it back unchanged in the reply to the server. This attribute is single-valued.

authStopMenuId

- Syntax: ces
- Description: Used internally by the RADIUS server.

authType

- Syntax: ces
- Description: Indicates to the RADIUS server how passwords are stored, so that the password supplied by the user can be compared correctly against the password stored under the user's entry in the directory. Possible values for this attribute are:
  - Crypt-Local — specifies that passwords are stored encrypted
  - Local — specifies that passwords are stored in clear text
  - System — specifies that passwords are maintained in `/etc/passwd`

authSuffixName

- Syntax: ces
- Description: Used internally by the RADIUS server to distinguish between the user name to process for authentication and a possible suffix. In some cases, the domain name can be added to the user's name, for example, jsmith@eng.xyz.com.

authTerminationAction

- Syntax: ces
- Description: Indicates the action to perform by the NAS when the service session is finished.

automountInformation

- Syntax: ces
- Description: The automount information for the entry in the NIS automount map.

bootFile

- Syntax: ces
- Description: The name of the file containing the boot parameters for the bootable device described by the entry.

bootParameter

- Syntax: ces
- Description: A boot parameter for the bootable device described by the entry.

buildingName

- Syntax: cis
- Description: The name of the building where the object described by the entry resides.

businessCategory

- Syntax: cis
- Description: The type of business of the object described by the entry.

c, see countryName

carLicense

- Syntax: cis
- Description: The type of car license held by the person described by the entry.

cACertificate

- Syntax: bin
- Description: The public key of the certification authority described by the entry.

cCMailAddress

- Syntax: cis
- Description: Used to route email messages through a Lotus CC:mail channel. It stores a copy of the email addresses in the preferredCCMailOriginator and preferredCCMailRecipient attributes.

certificateRevocationList

- Syntax: cis

- Description: A list of certificates that have been revoked by the certification authority described by the entry, or that the certification authority knows have been revoked by other certification authorities.

channelName

- Alternate name: ch
- Syntax: cis
- Description: The name of the Legacy Mail channel for the user described by the entry. Channel names are chosen for users by the system administrator.

channelType

- Syntax: ces
- Description: The type of the Legacy Mail channel for the user described in the entry. The value must be one of the following:
  - 0 for CC:mail
  - 1 for Microsoft Mail
  - 4 for an SMTP mail system
  - 8 for IBM PROFS

chapPassword

- Syntax: ces
- Description: Contains the response value provided by a PPP Challenge Handshake Authentication Protocol (CHAP) user in response to a challenge. This attribute is single-valued.

cn, see commonName

commonName

- Alternate name: cn
- Syntax: cis
- Description: The full name of the user described by the entry.

copyright

- Syntax: cis

- Description: The copyright statement for the object described by the entry.

countryName

- Alternate name: c
- Syntax: cis
- Description: The name of the country where the object described by the entry resides, or where a parent of the entry resides. The name has to be a two-letter string defined in ISO 3166. This attribute is single-valued. Multinational corporations usually use the country of their headquarters as the country of the whole organization.

createtimestamp

- Syntax: utctime
- Description: A timestamp that indicates the time at which the entry was created. This attribute is single-valued. It is created and maintained by the server.

creatorsName

- Syntax: dn
- Description: The DN of the person who created the entry. This attribute is single-valued. It is created and maintained by the server.

crossCertificatePair

- Syntax: cis
- Description: A pair of certificates, containing the public keys of the object described by the entry.

currentSequenceNumber

- Syntax: ces
- Description: A sequence number used during Legacy Mail directory synchronization.

dataSource

- Syntax: cis
- Description: The original data source or migration tool for data in the entry.

dc, see domainComponent

deltaRevocationList

- Syntax: cis
- Description: The differences in revocation lists. This attribute provides a list of newly revocated certificates.

departmentNumber

- Syntax: cis
- Description: A string identifying the department to which a user described by the entry belongs. The format is a local decision.

description

- Syntax: cis
- Description: The description of the entry object.

destinationIndicator

- Syntax: cis
- Description: The country and city addressing information for the object described by the entry.

dictionaryFile

- Syntax: ces
- Description: Specifies the dictionary to be used by the RADIUS server when it receives a request from the NAS described by the entry.

distinguishedName

- Alternate name: dn
- Syntax: dn
- Description: Specifies the distinguished name of an entry.

dITContentRules

- Syntax: cis

- Description: Specifies the rules governing the content of the DIT.

dITRedirect

- Syntax: dn
- Description: Indicates that the object described by one entry now has a newer entry in the DIT. The entry containing the redirection attribute should be removed after a suitable period.

dITStructureRules

- Syntax: cis
- Description: Specifies the rules governing the structure of the DIT.

dmdName

- Syntax: cis
- Description: Gives the name of the Directory Management Domain (DMD) stored on the server.

dn, see distinguishedName

dNSRecord

- Syntax: ces
- Description: Used to store DNS record fields.

docConvPreference

- Syntax: cis
- Description: The preferred method for converting a document sent through the gateway described by the entry.

documentAuthor

- Syntax: dn
- Description: The author of the document described by the entry.

documentIdentifier

- Syntax: cis
- Description: A string identifying the document described by the entry.

documentLocation

- Syntax: cis
- Description: The location of the document described by the entry.

documentPublisher

- Syntax: cis
- Description: The publisher of the document described by the entry.

documentStore

- Syntax: cis
- Description: The location where the document described by the entry is stored.

documentTitle

- Syntax: cis
- Description: The title of the document described by the entry.

documentVersion

- Syntax: cis
- Description: The version number of the document described by the entry.

domainComponent

- Alternate name: dc
- Syntax: cis
- Description: Part of the name of the domain described by the entry. This attribute is single-valued.

drink

- Alternate name: favouriteDrink

- Syntax: cis
- Description: The favorite drink of the person described by the entry.

dynamicIPaddressBinding

- Syntax: cis
- Description: When RADIUS accounting is activated, associates the dynamicIPAddress and the dynamicSessionId assigned to the remote user.

dynamicIPAddress

- Syntax: cis
- Description: When RADIUS accounting is activated, the IP address assigned to the remote user is recorded in the user's entry using this attribute. This attribute is created when the session begins, and removed when the session ends.

dynamicSessionCounter

- Syntax: int
- Description: When RADIUS accounting is activated, the number of concurrent open sessions for a remote user is recorded in the user's entry using this attribute. This attribute is removed when the user ends the last session. This attribute is single-valued.

dynamicSessionId

- Syntax: cis
- Description: When RADIUS accounting is activated, the session identifier assigned to the remote user for a particular session is recorded in the user's entry using this attribute. This identifier is used in to open and close the accounting report for the session.

employeeNumber

- Syntax: cis
- Description: A number identifying the person described by the entry.

employeeType

- Syntax: cis

- Description: Information identifying the type of the employee (for example, Contractor) described by the entry.

expandable

- Syntax: cis
- Description: Whether the membership of the list described by the entry is visible (TRUE or FALSE).

expirationDate

- Syntax: ces
- Description: Indicates the expiration date for the password stored in the userPassword attribute. The expirationDate attribute is single-valued.

facsimileTelephoneNumber

- Alternate name: fax
- Syntax: tel
- Description: The fax telephone number of the object described by the entry.

favouriteDrink, see drink

framedCompression

- Syntax: ces
- Description: Indicates a compression protocol to be used for the link.

framedIPAddress

- Syntax: ces
- Description: Indicates the address to be configured for the user.

framedMTU

- Syntax: ces
- Description: Indicates the maximum transmission unit (MTU) to be configured for the user, when it is not negotiated by some other means (such as PPP).

framedProtocol

- Syntax: ces
- Description: Indicates the framing to be used for framed access.

framedRoute

- Syntax: ces
- Description: Provides routing information to be configured for the user on the NAS. Not to be confused with the framedRouting attribute.

framedRouting

- Syntax: ces
- Description: Indicates the routing method for the user, when the user is a router to a network. Not to be confused with the framedRoute attribute.

freeFormName

- Syntax: cis
- Description: The name of the person described by the entry.

gecos

- Syntax: cis
- Description: The gecos field of the user described in the entry. Usually the user's common name. This attribute is single-valued.

generationQualifier

- Syntax: cis
- Description: Generation information, for example, Senior or III, to qualify the name of the user described by the entry.

gidNumber

- Syntax: long
- Description: An integer that uniquely identifies a group in an administrative domain. This attribute is single-valued.

givenName

- Alternate name: gn
- Syntax: cis
- Description: The given name of the person described by the entry.

grpCheckInfo

- Syntax: ces
- Description: Contains a list of attributes that must be checked by the RADIUS server against the information supplied by the remote user. If the grpCheckInfo attribute is not present, or if it does not contain any attributes, then all the attributes in the remote user's entry are checked before access is granted to the user. This attribute is used internally by the server.

grpReplyInfo

- Syntax: ces
- Description: Contains a list of attributes returned by the RADIUS server with an access-accept or access-reject response. It can contain connection parameters such as a PPP or SLIP profile. This attribute is used internally by the server.

homeDirectory

- Syntax: ces
- Description: The filesystem location of the home directory of the user described by the entry. This attribute is single-valued.

homeFacsimileTelephoneNumber

- Alternate name: homeFax
- Syntax: tel
- Description: The home fax number of the user described by the entry.

homePhone

- Syntax: tel
- Description: The home phone number of the user described by the entry.

homePostalAddress

- Syntax: cis
- Description: The home postal address of the user described by the entry.

host

- Syntax: cis
- Description: The host used by the object described by the entry.

idleTimeoutNumber

- Syntax: ces
- Description: Sets the maximum number of consecutive seconds that the connection can remain idle before the session is terminated.

initials

- Syntax: cis
- Description: The initials of the person described by the entry.

internationalISDNNumber

- Syntax: cis
- Description: The ISDN telephone number of the object described by the entry, including country and area codes.

ipHostNumber

- Syntax: cis
- Description: Specifies the IP address of the host described by the entry, in dotted decimal format, without leading zeros.

ipLoginHost

- Syntax: cis
- Description: Indicates the system with which to connect the user, when the authLoginService attribute is included in the connection request.

ipLoginPort

- Syntax: cis
- Description: Indicates the TCP port with which the user is to be connected, when the authLoginService attribute is included in the connection request.

ipNetmaskNumber

- Syntax: cis
- Description: Specifies an IP netmask, in dotted decimal format, without leading zeros. This attribute is single-valued.

ipNetworkNumber

- Syntax: cis
- Description: Specifies the number of the IP network described by the entry, in dotted decimal format, without leading zeros. This attribute is single-valued.

ipProtocolNumber

- Syntax: long
- Description: Specifies the port number/port type (for example UDP or TCP) pair for the IP protocol described by the entry. This attribute is single-valued.

ipxNetworkNumber

- Syntax: cis
- Description: Indicates the IPX network number to be configured for the user.

joinable

- Syntax: cis
- Purpose: Whether or not users may add themselves to the list described by the entry (TRUE or FALSE).

jpegPhoto

- Syntax: bin
- Description: A photograph, in JPEG format, of, or associated with, the object described by the entry.

keywords

- Syntax: cis
- Description: Keywords associated with the object described by the entry.

knowledgeInformation

- Syntax: cis
- Description: The knowledge information (references to other directory servers) stored by the DSA described by the entry.

labeledURI

- Alternate name: labeledURL
- Syntax: ces
- Description: The uniform resource identifier (URI) and label associated with the object described by the entry.

lastModifiedBy

- Syntax: dn
- Description: The distinguished name of the user who last modified the object described by the entry. Note that this is not the user who last modified the entry itself.

lastModifiedTime

- Syntax: utctime
- Description: The date and time when the object described by the entry was last modified. Note that this is not the date and time at which the entry itself was modified.

loginShell

- Syntax: ces
- Description: The path to the login shell of the user described by the entry. This attribute is single-valued.

ldapSyntaxes

- Syntax: cis

- Description: Specifies the syntaxes supported by the LDAP server.

localityName

- Alternate name: l
- Syntax: cis
- Description: The geographical locality of the object described by the entry.

lotusNotesAddresses

- Syntax: cis
- Description: The Lotus Notes electronic mail address of the user described by the entry.

macAddress

- Syntax: cis
- Description: Specifies a MAC address for the device described by the entry, expressed in colon-separated hex notation.

mail

- Alternate name: preferredRfc822Originator
- Syntax: cis
- Description: The advertised electronic mail address, in RFC822 format, of the user described by the entry.

mail11Addresses

- Syntax: cis
- Description: The Mail-11 electronic mail address of the user described by the entry.

mailAutoReplyExpirationDate

- Syntax: cis
- Description: At midnight on this date, disable auto-reply to email sent to the user described by the entry. The date must be in UTC format.

mailAutoReplyMode

- Syntax: cis
- Description: Mode of operation for the auto-reply facility (currently only vacation is supported) for the user described by the entry.

mailAutoReplySubject

- Syntax: cis
- Description: The subject line of an auto-reply message from the user described by the entry. If it contains the token $SUBJECT, it is replaced by the subject line of the incoming message.

mailAutoReplyText

- Syntax: cis
- Description: The body of an auto-reply message from the user described by the entry. If the text contains the tokens $SUBJECT or $BODY, they are replaced by the subject line or body from the incoming message. Use '$' as the line-separator.

mailAutoReplyTextInternal

- Syntax: cis
- Description: The body of an auto-reply message from the user described by the entry, for use within the organization. If the text contains the tokens $SUBJECT or $BODY, they are replaced by the subject line or body from the incoming message. Use '$' as the line-separator.

mailDeliveryFile

- Syntax: ces
- Description: The name of a file. Mail delivered to the user whose entry contains this attribute is appended to this file.

mailDeliveryOption

- Syntax: cis
- Description: One or more delivery options:
  - mailbox - deliver mail to the message store mailbox specified by the mailMessageStore attribute

- shared - deliver mail to the message store shared-mailbox specified by the mailMessageStore attribute
- native - deliver mail to a Unix filesystem mailbox
- autoreply - deliver mail to an auto-reply facility (for example, vacation mail)
- program - deliver mail to the Unix program specified by the mailProgramDeliveryInfo attribute
- forward - forward mail to the address specified by the mailForwardingAddress attribute
- file - append mail to the file specified by the mailDeliverFile attribute

Email received by the user described by the entry is delivered according to the options selected.

mailFolderMap

- Syntax: cis
- Description: The message store for a user's mail folders. The value must be one of the following:
  - UNIX V7 - UNIX V7 message store (also known as the `/var/mail` message store)
  - Sun-MS - Sun Internet Mail message store

mailForwardingAddress

- Syntax: cis
- Description: Forward mail received by the user described by the entry to the specified email address (RFC-822 format).

mailHost

- Syntax: cis
- Description: The hostname of the SMTP/MIME mail server of the user described by the entry, including the full domain name.

mailMessageStore

- Syntax: ces
- Description: The filesystem location for the inbox of the user described by the entry.

mailProgramDeliveryInfo

- Syntax: ces
- Description: One or more commands, with arguments, to be executed when a message is delivered to the user whose entry contains this attribute if the attribute mailDeliveryOptions contains the value program.

mailQuota

- Syntax: cis
- Description: The maximum size (in bytes) of the message store of the user described by the entry. A value of zero denotes an unlimited message store.

manager

- Syntax: dn
- Description: The distinguished name of the manager of the person or object described by the entry.

matchingRules

- Syntax: cis
- Description: Specifies the matching rules allowed in the schema.

matchingRuleUse

- Syntax: cis
- Description: Specifies how matching rules are used in the schema.

maxLastModifiedTime

- Syntax: cis
- Description: A timestamp used during Legacy Mail directory synchronization.

member

- Syntax: dn
- Description: The distinguished name of a member of the distribution list described by the entry.

memberUid

- Syntax: cis
- Description: The uid name of a member of the Posix group described by the entry.

memberNisNetgroup

- Syntax: cis
- Description: The name of a member of the netgroup described by the entry.

middleName

- Syntax: cis
- Description: The middle name of the person described by the entry.

mobiletelephoneNumber

- Alternate name: mobile
- Syntax: tel
- Description: The telephone number of the mobile phone used by the person described in the entry.

modifiersName

- Syntax: dn
- Description: The DN of the person who modified the entry . This attribute is single-valued.

modifytimestamp

- Syntax: utctime
- Description: A timestamp that indicates the time at which the entry was modified. This attribute is single-valued.

mrAddresses

- Syntax: cis
- Description: The address of a user of the MR mail system.

mSMailAddresses

- Syntax: cis
- Description: Used to route email messages through a Microsoft Mail channel. It stores a copy of the email addresses in the preferredMSMailOriginator and preferredMSMailRecipient attributes.

nameForms

- Syntax: cis
- Description: Specifies the name forms allowed in the schema.

namingContexts

- Syntax: dn
- Description: Specifies the master and slave naming contexts stored on the server.

nGM70Addresses

- Syntax: cis
- Description: The electronic mail address of a user of the NMG70 mail system.

nGMAddresses

- Syntax: cis
- Description: The electronic mail address of a user of the NMG mail system.

nisMapEntry

- Syntax: ces
- Description: Contains a record in the NIS map described by the entry. This attribute is single-valued.

nisMapName

- Syntax: ces
- Description: Specifies the name of the NIS map described by the entry. This attribute is single-valued.

nisNetgroupTriple

- Syntax: cis
- Description: Represents a triple of the form hostname/username/domainname associated with the netgroup described by the entry.

nisNetIdGroup

- Syntax: ces
- Description: Represents the group id associated with a record in the netid.byname map.

nisNetIdHost

- Syntax: ces
- Description: Represents the hostname associated with a record in the netid.byname map.

nisNetIdUser

- Syntax: ces
- Description: Represents the user id associated with a record in the netid.byname map.

o, see organizationName

objectClass

- Syntax: cis
- Description: The object class of the type of entry.

objectClasses

- Syntax: cis
- Description: Specifies the object classes allowed in the schema.

objectStatus

- Syntax: cis

- Description: Used during Legacy Mail directory synchronization to denote a deleted entry.

obsoletedByDocument

- Syntax: cis
- Description: Information identifying a document that makes the document described in the entry obsolete.

obsoletesDocument

- Syntax: cis
- Description: Information identifying a document that is made obsolete by the document described in the entry.

oncRpcNumber

- Syntax: long
- Description: RPC number of the RPC service described by the entry. This attribute is single-valued.

organizationName

- Alternate name: o
- Syntax: cis
- Description: The name of the organization to which the object described by the entry belongs.

owner

- Syntax: dn
- Description: The distinguished name of an entry describing the person responsible for the distribution list described by the entry.

ownerDeliveryFile

- Syntax: ces
- Description: The name of the file to which mail addressed to the owner of the distribution list described by the entry is appended.

ownerDeliveryOption

- Syntax: ces
- Description: Delivery options for mail addressed to owner-*listname*. The values must be one of the following:
  - mailbox: deliver mail to a message store mailbox
  - shared: deliver mail to a message store shared mailbox
  - native: deliver mail to a UNIX filesystem mailbox
  - autoreply: deliver mail to the autoreply facility, such as a vacation mailer
  - program: deliver mail to the UNIX program specified as the value of the attribute ownerProgramDeliveryInfo
  - forward: forward mail to another recipient
  - file: append mail to the file specified as a value of the attribute ownerDeliveryFile

ownerProgramDeliveryInfo

- Syntax: ces
- Description: Mail addressed to the owner of a distribution list is delivered to this program. Specifies one or more commands, with arguments, to use in program delivery. Use '$' as the line-separator.

pagertelephonenumber

- Alternate name: pager
- Syntax: tel
- Description: The telephone number of the pager of the person described by the entry.

personalMobile

- Syntax: tel
- Description: The telephone number of the mobile phone belonging personally to the person described by the entry.

personalPager

- Syntax: tel
- Description: The telephone number of the pager belonging personally to the person described by the entry.

personalSignature

- Syntax: bin
- Description: The signature of the person described by the entry.

personalTitle

- Syntax: cis
- Description: The title of the person described by the entry, for example, Doctor, or Ms.

photo

- Syntax: bin
- Description: A photograph of, or associated with, the object described by the entry.

physicalDeliveryOfficeName

- Syntax: cis
- Description: The mailstop of the object described by the entry.

postalAddress

- Syntax: cis
- Description: The postal address of the object described by the entry.

postalCode

- Syntax: cis
- Description: The postal code of the object described by the entry.

postOfficeBox

- Syntax: cis
- Description: The post office box of the object described by the entry.

preferredCCMailOriginator

- Syntax: cis

- Description: The email address for routing through a Lotus CC:mail channel.

preferredCCMailRecipient

- Syntax: cis
- Description: The native Lotus CC:mail address.

preferredDeliveryMethod

- Syntax: cis
- Description: Preferred delivery method for communication with the object described by the entry. This attribute is single-valued.

preferredLanguage

- Syntax: cis
- Description: The preferred language for communication with the person described by the entry. This attribute is single-valued.

preferredLotusNotesOriginator

- Syntax: cis
- Description: The email address used for routing through a Lotus Notes channel.

preferredLotusNotesRecipient

- Syntax: cis
- Description: The native Lotus Notes mail address.

preferredMail11Originator

- Syntax: cis
- Description: The email address used for routing through a Mail-11 channel.

preferredMail11Recipient

- Syntax: cis
- Description: The native Mail-11 mail address.

preferredMrOriginator

- Syntax: cis
- Description: The email address used for routing through a Mail Relay (MR) channel.

preferredMrRecipient

- Syntax: cis
- Description: The native MR mail address.

preferredMSMailOriginator

- Syntax: cis
- Description: The email address for routing through a Microsoft Mail channel.

preferredMSMailRecipient

- Syntax: cis
- Description: The native Microsoft Mail address.

preferredNGM70Originator

- Syntax: cis
- Description: The email address used for routing through a Novell Groupwise Mail 7.0 (NMG70) channel.

preferredNGM70Recipient

- Syntax: cis
- Description: The native NMG70 address.

preferredNGMOriginator

- Syntax: cis
- Description: The email address used for routing through a Novell Groupwise Mail (NMG) channel.

preferredPROFSOriginator

- Syntax: cis

- Description: The email address for routing through an IBM PROFS channel.

preferredPROFSRecipient

- Syntax: cis
- Description: The native IBM PROFS address.

preferredRfc822Recipient

- Syntax: cis
- Description: The user's internal email address (RFC-822 format).

presentationAddress

- Syntax: cis
- Description: The presentation address of the object described by the entry. This attribute is single-valued.

pROFSAddresses

- Syntax: cis
- Description: Used to route email messages through an IBM PROFS channel. It stores a copy of the email addresses in the preferredPROFSOriginator and preferredPROFSRecipient attributes.

radiusAuthFailedAccess

- Syntax: ces
- Description: Created dynamically in a remote user's entry when an access request is rejected. This counter is incremented by 1 at each failed attempt. The user account is blocked when this counter reaches the blocking value specified in the configuration (by default, 4). This attribute is single-valued.

radiusLoginExpiration

- Syntax: ces
- Description: Indicates the expiration date for the password stored in the radiusLoginPasswd attribute. The radiusLoginExpiration attribute is single-valued.

radiusLoginPasswd

- Syntax: ces
- Description: Password provided by the remote user to gain access to the network through the LOGIN protocol. This attribute is single-valued.

radiusLoginProfile

- Syntax: ces
- Description: Flag with value 0 or 1. Value 1 indicates to check the password supplied by the user against the password stored in the radiusLoginPasswd attribute. Value 0 disables this check. The radiusLoginProfile attribute is single-valued.

radiusPppExpiration

- Syntax: ces
- Description: Indicates the expiration date for the password stored in the radiusPppPasswd attribute. The radiusPppExpiration attribute is single-valued.

radiusPppPasswd

- Syntax: ces
- Description: Password provided by the remote user to gain access to the network through the PPP protocol This attribute is single-valued.

radiusPppProfile

- Syntax: ces
- Description: Flag with value 0 or 1. Value 1 indicates to check the password supplied by the user against the password stored in the radiusPppPasswd attribute. Value 0 disables this check. The radiusPppProfile attribute is single-valued.

radiusSlipExpiration

- Syntax: ces
- Description: Indicates the expiration date for the password stored in the radiusSlipPasswd attribute. The radiusSlipExpiration attribute is single-valued.

radiusSlipPasswd

- Syntax: ces
- Description: Password provided by the remote user to gain access to the network through the SLIP protocol. This attribute is single-valued.

radiusSlipProfile

- Syntax: ces
- Description:Flag with value 0 or 1. Value 1 indicates to check the password supplied by the user against the password stored in the radiusSlipPasswd attribute. Value 0 disables this check. The radiusSlipProfile attribute is single-valued.

ref

- Syntax: ces
- Description: Provides a pointer to a subtree or entry in the DIT stored on a different data store. The pointer is a URL of the form "ldap://*hostname*[:*port*]/*DN*", where *hostname* is the name of the host where the data store resides, *port* is the LDAP port number (by default port 389), and *DN* is the data store suffix.

registeredAddress

- Syntax: cis
- Description: The registered postal address of the entity described by the entry.

reportsTo

- Syntax: cis
- Description: The name of the manager of the user described by the entry.

requestsToDeliveryFile

- Syntax: ces
- Description: The name of a file to which requests to be added to the distribution list described by the entry are appended.

requestsToDeliveryOption

- Syntax: cis

- Description: One or more delivery options for mail addressed to *listname*-request:
  - `mailbox` - deliver mail to a message store mailbox
  - `shared` - deliver mail to a message store shared-mailbox
  - `native` - deliver mail to a UNIX filesystem mailbox
  - `autoreply` - deliver mail to an auto-reply facility (for example, vacation mail)
  - `program` - deliver mail to a UNIX program
  - `forward` - forward mail to another address
  - `file` - append mail to a file

requestsToProgramDeliveryInfo

- Syntax: ces
- Description: Mail addressed to *listname*-request is delivered to this program. Specifies one or more commands, with arguments, to use in program delivery. Use $ as the line-separator.

rfc822AuthorizedSubmitter

- Syntax: cis
- Description: The email addresses of users authorized to post to the list.

rfc822Mailbox

- Syntax: cis
- Description: Stores all the email addresses (RFC-822 format) defined for the user. It stores a copy of the email addresses in the mail and preferredRfc822Recipient attributes.

rfc822MailMember

- Syntax: ces
- Description: Stores the email addresses (RFC-822 format) defined for members of the list.

rfc822Owner

- Syntax: cis
- Description: The email address of the owner of the list.

rfc822UnauthorizedSubmitter

- Syntax: cis
- Description: The email addresses of users not authorized to post to the list.

roleOccupant

- Syntax: cis
- Description: Information identifying the object or person fulfilling the role described by the entry.

roomNumber

- Syntax: cis
- Description: The number of the room where the object described by the entry is located.

searchGuide

- Syntax: cis
- Description: Information to facilitate searching for information contained in the entry.

secretary

- Syntax: dn
- Description: The distinguished name of the secretary of the person or organization described by the entry.

seeAlso

- Syntax: dn
- Description: The distinguished name of an entry that contains information that is also of interest to anyone interested in the object described by this entry.

serialNumber

- Syntax: cis
- Description: The serial number of the device described by the entry.

sessionTimeoutNumber

- Syntax: ces
- Description: Sets the maximum number of seconds of service to be provided to the user described in the entry before the session is shut down.

shadowLastChange

- Syntax: long
- Description: Indicates the number of days between January 1, 1970 and the day when the user password was last changed in the `/etc/shadow` file. This attribute is single-valued.

shadowExpire

- Syntax: long
- Description: Indicates the date on which the user login will be disabled. This attribute is single-valued.

shadowFlag

- Syntax: long
- Description: Reserved attribute, not currently in use.

shadowInactive

- Syntax: long
- Description: Indicates the number of days of inactivity allowed for the user. This attribute is single-valued.

shadowMax

- Syntax: long
- Description: Indicates the maximum number of days for which the user password remains valid. This attribute is single-valued.

shadowMin

- Syntax: long
- Description: Indicates the minimum number of days required between password changes. This attribute is single-valued.

shadowWarning

- Syntax: long
- Description: The number of days of advance warning given to the user before the user password expires. This attribute is single-valued.

sharedKey

- Syntax: ces
- Description: Specifies the shared secret used by the network access server (NAS) described by the entry during RADIUS authentication. This attribute is single-valued.

stateOrProvinceName

- Alternate name: st
- Syntax: cis
- Description: The name of the state, province, or geographical area within a country where the object described by the entry resides.

streetAddress

- Alternate name: street
- Syntax: cis
- Description: A street name and number.

subject

- Syntax: cis
- Description: The subject of the document described by the entry.

subschemaSubentry

- Syntax: dn
- Description: The DN of the subschema entry or subentry that contains the attributes specifying the schema. This attribute is single-valued.

sunNisDbmCache

- Syntax: cis

- Description: Indicates whether the NIS/LDAP server must maintain the map described by the entry in plain NIS format. The possible values of this attribute are Enabled or Disabled. It is usually Enabled for a master server, and Disabled for a slave server. The value automatically changes to Disabled for a map that exceeds 50 000 entries. This attribute is single-valued.

sunNisDnsForwarding

- Syntax: cis
- Description: Indicates that the server must look up DNS for hostnames and addresses not found in the NIS tables. The possible values of this attribute are Enabled or Disabled. This attribute is created with the value Enabled when you run the `dsypinit` command with option `-b`. This attribute is single-valued.

sunNisDomain

- Syntax: ces
- Description: Gives the name of the NIS domain to which the map described by the entry belongs. This attribute is single-valued.

sunNisInputFile

- Syntax: ces
- Description: Stores the value of a special NIS key called `YP_INPUT_FILE`. This attribute is single-valued.

sunNisKey

- Syntax: ces
- Description: Stores the value of a NIS key (case-sensitive).

sunNisLoadMap

- Syntax: cis
- Description: Adding this attribute to the entry launches a reload of the map described by the entry. It builds the map from the entries already present in the directory. This attribute is single-valued, and you can give it any value. This attribute is automatically removed when the reload of the map is complete. Creating this attribute is equivalent to running the `dsypinit` command with option `-l`.

sunNisMapFullName

- Syntax: ces
- Description: Gives the full name of a Sun NIS map with the domain name as suffix. This attribute is single-valued.

sunNisMapState

- Syntax: cis
- Description: Indicates whether the map described by the entry is supported by the server. The possible values of this attribute are Enabled and Disabled. Enabled indicates that the map is supported by the server, Disabled that it is not. This attribute is single-valued.

sunNisMaster

- Syntax: ces
- Description: Specifies the hostname of the master server for the map described by the entry. This attribute is single-valued.

sunNisOutputName

- Syntax: ces
- Description: Stores the value of a special NIS key called `YP_OUTPUT_FILE` . This attribute is single-valued.

sunNisSecurityMode

- Syntax: cis
- Description: Sets the security mode for the map described by the entry. The possible values of this attribute are Secure and Insecure. When set to Secure, the server will accept connections from secure networks only. This attribute is single-valued. Setting this attribute to Secure is equivalent to running the `dsypinit` command with the option `-r`.

supportedAlgorithms

- Syntax: cis
- Description: Specifies the algorithms that the server supports.

supportedApplicationContext

- Syntax: cis
- Description: An application context supported by the application entity described by the entry.

supportedControl

- Syntax: cis
- Description: Specifies the LDAP v3 controls that the server supports.

supportedLDAPVersion

- Syntax: int
- Description: Specifies the LDAP version that the server supports.

supportedExtension

- Syntax: cis
- Description: Specifies the LDAP v3 extensions that the server supports.

supportedSASLMechanisms

- Syntax: cis
- Description: Specifies the Simple Authentication Security Layer (SASL) mechanisms that the server supports.

surname

- Alternate name: sn
- Syntax: cis
- Description: The surname of the person described by the entry.

telephoneNumber

- Syntax: tel
- Description: Telephone number (in international format).

teletexTerminalIdentifier

- Syntax: cis
- Description: The teletex terminal identifier and, optionally, parameters for a teletex terminal associated with the object described by the entry.

telexNumber

- Syntax: cis
- Description: Telex number, country code and answerback code of a telex terminal. Dollar($)-separated string.

textEncodedORaddress

- Syntax: cis
- Description: The X.400 electronic mail originator/recipient address (ORAddress) of the user described in the entry.

thumbNailPhoto

- Syntax: bin
- Description: A thumbnail photograph of, or associated with, the object described by the entry.

thumbNailLogo

- Syntax: bin
- Description: A thumbnail logo associated with the object described by the entry.

title

- Syntax: cis
- Description: The title of the person described by the entry, for example, Doctor, or Ms.

uidNumber

- Syntax: long
- Description: An integer that uniquely identifies a user in an administrative domain. This attribute is single-valued.

unauthorizedDomain

- Syntax: cis
- Description: A domain name from which users are not authorized to post to the list.

unauthorizedSubmitter

- Syntax: cis
- Description: The registered users not authorized to post messages to the list.

uniqueIdentifier

- Syntax: cis
- Description: A unique identifier for the object described by the entry.

uniqueMember

- Syntax: cis
- Description: A unique member of the group described by the entry.

updatedByDocument

- Syntax: cis
- Description: Information identifying a document that updates the document described by the entry.

updatesDocument

- Syntax: cis
- Description: Information identifying a document that is updated by the document described by the entry.

userCallbackId

- Syntax: ces
- Description: Indicates a name of a place to be called. This attribute is interpreted by the NAS.

userCallbackNumber

- Syntax: ces
- Description: Indicates a dialing string to use for callback to provide service to the user.

userCertificate

- Syntax: cis
- Description: A certificate containing the public key of the user described by the entry.

userDefinedAttribute1

- Syntax: cis
- Description: Attribute for use by the user.

userDefinedAttribute2

- Syntax: cis
- Description: Attribute for use by the user.

userDefinedAttribute3

- Syntax: cis
- Description: Attribute for use by the user.

userDefinedAttribute4

- Syntax: cis
- Description: Attribute for use by the user.

userid

- Alternate name: uid
- Syntax: cis
- Description: The user ID of the user described by the entry.

userPassword

- Syntax: protected
- Description: The password that the user described by the entry uses to gain access to the entry.

userSMIMECertificate

- Syntax: bin
- Description: The Secure MIME certificate of the user described by the entry.

x121Address

- Syntax: cis
- Description: An address as defined by ITU Recommendation X.121.

x500uniqueIdentifier

- Syntax: cis
- Description: A unique identifier for the object described by the entry.

CHAPTER **9**

Implementing Replication

This chapter explains how replication works in Sun Directory Services, and explains how to configure replication for your directory servers using the Admin Console.

How Replication Works

Information from a master naming context is propagated to a replica by the `dspushd` daemon. This information can also be *pulled* from the master server by the `dspulld` daemon running on the replica server. The `dspushd` and `dspulld` daemons use the LDAP protocol to update a replica naming context.

A master naming context for which a replica is defined maintains a replication log. Each time the master naming context is updated, the transaction is recorded in the replication log. When the `dspushd` daemon next runs, it reads the replication log and sends the change to the server that holds the replica naming context. The `dsservd` server handles update requests from `dspushd` in the same way that it handles all requests, using the information supplied in the bind request to set the access level granted to `dspushd` requests. To guarantee that all replication updates are completed, `dspushd` must bind with the DN defined when the replica naming context was configured. If a different DN is used, write access for all entries may not be granted.

If replication is managed by the slave server, the `dspulld` daemon compares the master and the replica naming contexts, and performs the necessary updates on the replica.

A replica data store always has a referral pointing to the master data store. If a replica server receives a request to modify an entry, it returns a referral to the client, indicating the master server to be contacted. Once the modification has been made in the master naming context, the change is sent to the replica naming context the next time the `dspushd` daemon runs.

Any naming context held in the data store, including replica naming contexts, can be replicated to another server.

Setting Up Replication

Before you start configuring replication on master servers and slave servers, you must make the following decisions:

- What information you want to replicate
- Which server will hold the master copy
- Which servers will hold replicas
- Will the master server push updates to all slaves, or will replica servers individually pull updates from the master

In defining the information you want to replicate, you specify:

- The entries you want to replicate
- The attributes you want to replicate

To select the entries to replicate, you can:

- Provide the DN of a subtree: all the entries in the subtree are replicated
- Specify an individual entry (object)
- Specify a filter

To select the attributes to replicate, you can:

- Specify all attributes
- Include or exclude certain attributes from the replication

You can also define a replication synchronization schedule. This schedule determines when all updates are sent to replicas. There are three types of synchronization:

- *Immediate*, which means that the replication daemon, `dspushd`, runs permanently and sends updates to the replica immediately when modifications are made in the master.
- *Delayed*, which means that modifications are logged until the next time `dspushd` runs. If you select Delayed synchronization, specify a schedule for `dspushd`.
- *Disabled*, which means that modifications are not automatically sent to the replica.

You can use the *Send updates now* control in the Admin Console to send any outstanding modification immediately to a replica. Setting the synchronization type to *Disabled* and using the *Send updates now* control to initiate replication updates manually can be useful where the update traffic is unpredictable, or where the remote server is connected by a dial-up line.

Operations to Perform on the Master Server

On the master server you must:

- Declare all the replica servers — see “To Create Replicas” on page 197
- Optionally, set up a synchronization schedule — see “To Set Up a Replication Synchronization Schedule for a Master Server” on page 199

To Create Replicas

1. **In the Create Data Store window or the Modify Data Store window, choose Replica from the Create menu.**

   The Add Replica window is displayed.

2. **Specify the type of replica (subtree or object).**

3. **In the Subtree/Entry field, specify the distinguished name of the subtree or object to be replicated.**

   To create a replica of the whole data store, specify the DN of the naming context used to identify the data store.

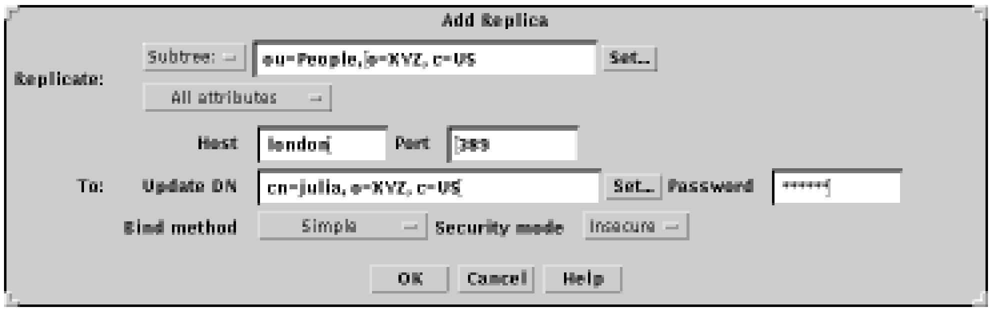
4. **Select the attributes to be replicated.**

   You can specify that all attributes are replicated, or you can exclude or include certain attributes. If you choose Exclude or Include from the Attributes pop-up menu, specify the particular attributes you want to exclude from or include in the replica.

5. **Specify the following parameters for the target:**

   - The name of the target host (replica server) where the replica will be stored
   - Specify the port of the `dsservd` server on the replica server to be used by the replication daemon, `dspushd`
   - The distinguished name and password that the master will supply when requesting authentication
   - The Bind method, simple, SASL with CRAM-MD5, or SASL with the EXTERNAL mechanism
   - The security mode, insecure, TLS, or SSL, and the SSL key package if you select TLS or SSL as the security mode

   For example, Figure 9–1 shows an example of replicating the naming context ou=People, o=XYZ, c=US from the boston server (not shown) to the london slave server.


Add Replica

Replicate: Subtree: ou=People, o=XYZ, c=US Set...

All attributes

To: Host london Port 389

Update DN cn=julia, o=XYZ, c=US Set... Password ******

Bind method Simple Security mode Insecure

OK Cancel Help

Figure 9–1 Add Replica Window

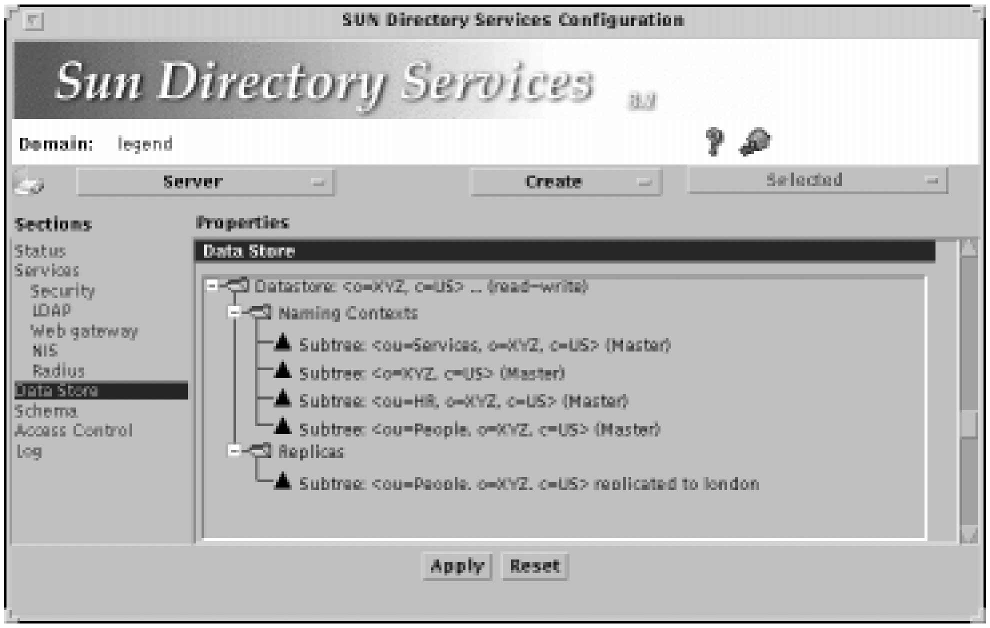
6. **Click OK to save the replica definition and exit from the Add replica window.**

   Information concerning the replica is added to the list in the Replicas section of the Create Data Store or Modify Data Store window.

7. **Click OK to dismiss the data store window.**

   In the Data Store section of the Admin Console main window, click on the folder icon for the data store to check that the replica you have just created is listed under the Replicas folder.

   For example, Figure 9–2 shows the naming contexts held on the boston server, and the replica defined for the london slave server.



```
Datastore: <o=XYZ, c=US> ... (read-write)
  Naming Contexts
    Subtree: <ou=Services, o=XYZ, c=US> (Master)
    Subtree: <o=XYZ, c=US> (Master)
    Subtree: <ou=HR, o=XYZ, c=US> (Master)
    Subtree: <ou=People, o=XYZ, c=US> (Master)
  Replicas
    Subtree: <ou=People, o=XYZ, c=US> replicated to london
```

Figure 9–2 Data Store Contents

To Set Up a Replication Synchronization Schedule for a Master Server

1. **Go to the Data Store section of the Admin Console.**

2. **From the Send Updates to Replica menu button select Delayed.**
   A Day menu button and Time menu button are displayed.

3. **Select the day and time at which you want replication to occur periodically.**

4. **Click Apply in the Admin Console main window to save your changes.**

To Start Replication At Any Time from a Master Server

♦ **In the Data Store section of the Admin Console main window, highlight the replica that you want to synchronize with the master, and click the Send updates now button.**

Operations to Perform on Slave Servers

On your slave servers, you must:

- Create the naming context for the replica on the server, as described in “To Create a Data Store” on page 44, making sure that you select Slave mode
- Optionally, configure pull replication, if you have not configured the master to push updates automatically to replicas
- Optionally, set up a synchronization schedule — see “To Start Replication At Any Time from a Slave Server” on page 201

Note - You must ensure that the schema supported by the slave server is compatible with the entries that you want to replicate. If entries to replicate depend on changes made to the master server schema, you must make the same changes to the slave server schema. It is not sufficient to copy the subschema object class from the master server schema to the slave server schema.

To Configure Pull Replication

1. **Create the replica naming context.**

This procedure is described as part of the procedure for creating a data store in “To Create a Data Store” on page 44. If the replica naming context exists, double-click on it in the Data Store section of the Admin Console.

1. **In the Add Naming Context Window or the Modify Naming Context Window, set the Configure Pull Replication button to Yes.**

   This displays additional replication parameter fields.

2. **Select the attributes to replicate.**

   If you select Include attributes or Exclude attributes, you must specify a list of attributes to include in or exclude from the replication.

3. **Specify the following parameters for the master server:**
   - The distinguished name and password that the slave server will supply when requesting authentication
   - The LDAP timeout on the bind request
   - The Bind method, security mode, and SSL key package if you select SSL as the security mode

4. **Click OK to save your changes and dismiss the Add Naming Context window.**

In the Data Store section of the Admin Console main window, an extra set of controls is displayed. These controls enable you to set up a synchronization schedule on the slave server, and to start replication at any time from the slave.

To Set Up a Replication Synchronization Schedule for a Slave Server

1. **Go to the Data Store section of the Admin Console.**
2. **From the Request Updates from Master menu button select Enabled.**

   A Day menu button and Time menu button are displayed.
3. **Select the day and time at which you want replication to occur periodically.**
4. **Click Apply in the Admin Console main window to save your changes.**

   This enables the `dspulld` command.

To Start Replication At Any Time from a Slave Server

♦ **In the Data Store section of the Admin Console main window, highlight the replica that you want to synchronize with the master, and click the Request updates now button.**

Initializing Replication

After you have configured a replica naming context, the master and replica data stores must be in the same state, so that the replica can receive replication updates from the master. If the master data store already contains entries, the Admin Console displays a dialog box giving you the option of populating the replica. Use this facility to populate the replica automatically with the entries that the master contains.

Note - Although you can start the replication process from the Admin Console, the Admin Console does not control the process and does not display error messages. You need to check the replication log files, `dspush.log` and `dspull.log`, to ensure that the replication process has completed successfully.

CHAPTER **10**

Managing the Directory Services

This chapter describes the tasks that you occasionally need to perform to manage your directory service.

Starting the Sun Directory Services

To start the directory server from the Admin Console, choose Start from the LDAP Server menu. You can also start the directory server daemon, `dsservd`, by typing the following command as `root`:

```
# /etc/init.d/dsserv start
```

To start the web gateway, `dswebd`, as `root` type:

```
# /etc/init.d/dsweb start
```

To start the directory administration server, `dsadm`, as `root` type:

```
# /etc/init.d/dsadm start
```

The RADIUS server daemon, `dsradiusd`, is started at the same time as the `dsservd` daemon. If you have stopped it independently from `dsservd`, you can also start it independently. As `root` type:

```
#/opt/SUNWconn/ldap/sbin/dsradius start
```

When you install the Sun Directory Services, these commands are added to the system startup file, so that all the server daemons are started automatically when the machine is rebooted.

Stopping the Sun Directory Services

To stop the directory server, from the Admin Console, choose Stop from the LDAP Server menu. You can also stop the directory server by typing the following command as `root`:

```
# /etc/init.d/dsserv stop
```

Stopping the directory server automatically stops the replication server. If you have set up a replication schedule, the replication server is restarted automatically when you restart the directory server, and will continue to follow the schedule.

To stop the web gateway, `dsweb`, as `root` type:

```
# /etc/init.d/dsweb stop
```

To stop the directory administration server, `dsadm`, as `root` type:

```
# /etc/init.d/dsadm stop
```

The RADIUS server daemon, `dsradiusd`, is stopped at the same time as the `dsservd` daemon. If you want to stop it independently from `dsservd`, as `root` type:

```
# /opt/SUNWconn/ldap/sbin/dsradius stop
```

Monitoring Directory Services with SNMP

Sun Directory Services provides two SNMP agents:

- `dsnmpserv`, to monitor the directory server
- `dsnmprad`, to monitor the RADIUS server

The first SNMP agent, `dsnmpserv`, supports the management information bases (MIBs) defined in the following standards:

- *Network Services Monitoring MIB* (RFC 1565)
- *X.500 Directory Monitoring MIB* (RFC 1567)

These MIBs are part of the messaging and directory management (MADMAN) standards that apply to all messaging and directory applications.

The second SNMP agent, `dsnmprad`, supports the MIBs defined in the following draft standards:

- RADIUS Authentication Server MIB (*draft-ietf-radius-auth-servmib-01.txt*)
- RADIUS Accounting Server MIB (*draft-ietf-radius-acc-servmib-01.txt*)

For a detailed list of the type of information collected by each agent, refer to "Directory Server Statistics" on page 206 and "RADIUS Server Statistics" on page 208.

On a Solaris 2.6 machine, the SNMP agents are started automatically during the installation process. This is possible because the Solaris 2.6 operating environment includes a master SNMP agent, `snmpdx` that resides on UDP port 161 and relays SNMP traffic to and from all other SNMP agents installed on the machine.

Starting and Stopping the SNMP Agents

If you need to start or stop an agent manually, use the following commands.

To start the SNMP agent for the directory server, `dsnmpserv`, as `root` type:

```
# /etc/init.d/init.dsnmpserv start
```

To start the SNMP agent for the RADIUS server, `dsnmprad`, as `root` type:

```
# /opt/SUNWconn/ldap/sbin/init.dsnmprad start
```

To stop the SNMP agent for the directory server, `dsnmpserv`, as `root` type:

```
# /etc/init.d/init.dsnmpserv stop
```

To stop the SNMP agent for the RADIUS server, `dsnmprad`, as `root` type:

```
# /opt/SUNWconn/ldap/sbin/init.dsnmprad stop
```

Configuring the SNMP Agents

When you install Sun Directory Services on a Solaris 2.6 machine, configuration information for the `dsnmpserv` and `dsnmprad` agents is added to the configuration of the Solaris master agent `snmpdx`, and all SNMP agents are started. By default, the Sun Directory Services agents report events to the local host. The UDP ports that the agents use are dynamically assigned by the master agent.

In both cases, you can configure the hosts to which the SNMP agents report events. This is done using the `dsnmpcfg` command as follows (you must be logged in as `root`):

```
# /opt/SUNWconn/ldap/sbin/dsnmpcfg configure
```

You are prompted to provide the hostnames of the machines to which you want each agent to report events. If the agents are running when you perform the configuration, they are restarted to take your changes into account.

Directory Server Statistics

This section lists the information collected by the directory server SNMP agent, `dsnmpserv`, and explains how to display it using the Admin Console.

Information Collected by `dsnmpserv`

The information collected by the dsnmpserv SNMP agent can be monitored from a management platform such as SunNet Manager™ or Solstice Enterprise Manager™. The following directory service information is monitored:

- Application information
  - Application name
  - Application directory name
  - Application version
  - Application uptime
  - Application status (*up* or *down*)
  - Last status change
  - Number of inbound associations
  - Number of outbound associations
  - Accumulated inbound associations
  - Accumulated outbound associations
  - Last inbound activity
  - Last outbound activity
  - Rejected inbound association
  - Failed inbound association
- Association information
  - The distinguished name of the remote application
  - The protocol being used to communicate
  - The type of the remote application, and whether it is an initiator or responder
  - The current duration of the association
- Directory server operations
  - Anonymous bind
  - Unauthenticated bind
  - Simple authentication bind

  - Strong authentication bind
  - Bind security errors
  - Inbound operations
  - Read operations
  - Compare operations
  - Add Entry operations
  - Delete Entry operations
  - Modify Entry operations
  - List operations
  - Search operations
  - One-level search operations
  - Whole tree search operations
  - Referrals
  - Chaining
  - Security errors
  - DSA errors

- Directory entry information

  - Master entries
  - Copy entries
  - Cached entries
  - Cache Hits
  - Slave Hits

- Interactions with other directory servers

  - Distinguished name of remote directory server
  - Time of creation of remote directory server
  - Time of last attempt to contact the remote directory server
  - Time of last successful interaction with the remote directory server
  - Number of failures since last successful contact
  - Total number of failures to contact the remote directory server
  - Total number of successful interactions with the remote directory server

Information Collected by `dsservd`

You can view statistics collected by the directory server, `dsservd`, in five categories:

- Global
- Detailed

- Operations
- Associations
- Interactions

  The statistics available are the same information that is collected by the `dsnmpserv` SNMP agent. See "Monitoring Directory Services with SNMP" on page 204 for details of the information collected.

To Monitor Directory Server Statistics

1. **In the Admin Console main window, go to the LDAP section under Services**
2. **Click the Show Statistics button.**

The LDAP Statistics window is displayed. It presents a snapshot of the statistics available for the directory server.

1. **Click the tab for the category that you want to view.**
2. **Click the Update button to get the latest statistics.**

   To update the statistics at regular intervals:

   a. **Set the Refresh Interval field**

   b. **Click Start Auto Update**

   Note - The Start Auto Update and Stop Auto Update controls apply to viewing the statistics, not to collecting the data. They only apply while the window is displayed. If you close the window, the refresh interval is reset to the default and automatic updating of the statistics view stops.

RADIUS Server Statistics

This section lists the information collected by the RADIUS server SNMP agent `dsnmprad`. This information can be monitored from a management platform such as SunNet Manager or Solstice Enterprise Manager.

Information Collected

The following RADIUS authentication service information is monitored:

- Server identifier
- Uptime
- Reset time
- Configuration reset
- Total access requests
- Total invalid requests
- Total duplicate access requests
- Total access requests
- Total access accepts
- Total access rejects
- Total access challenges
- Total malformed access requests
- Total bad authenticators
- Total packets dropped
- Total unknown type
- Client entry (contains authentication information monitored for every NAS connected to the server)
  - Client Index
  - Client Address
  - Client ID
  - Access requests
  - Duplicate access requests
  - Access accepts
  - Access rejects
  - Access challenges
  - Malformed access requests
  - Bad authenticators
  - Packets dropped
  - Unknown type

The following RADIUS accounting service information is monitored:

- NAS identifier
- Uptime
- Reset time

- Configuration reset
- Total requests
- Total invalid requests
- Total duplicate requests
- Total responses
- Total malformed requests
- Total bad authenticators
- Total packets dropped
- Total no record
- Total unknown type
- Client entry (contains accounting information monitored for every NAS connected to the server)
  - Client Index
  - Client Address
  - Client ID
  - Packets dropped
  - Requests
  - Duplicate requests
  - Responses
  - Bad authenticators
  - Malformed requests
  - No record
  - Unknown type

Displaying RADIUS Server Statistics

You cannot display RADIUS server statistics in the Admin Console. You need a management application such as SunNet Manager or Solstice Enterprise Manager. The files required to interoperate with these management applications are provided with Sun Directory Services:

- The directory `/opt/SUNWconn/ldap/snmp/snm` contains all files necessary for `dsnmprad` to report events to a SunNet Manager station
- The directory `/opt/SUNWconn/ldap/snmp/sem` contains all files necessary for `dsnmprad` to report events to a Solstice Enterprise Manager station

The Web Gateway

The web gateway provides an interface to an LDAP directory from any web browser. You can use this interface to browse the directory, to search for and read entries, and to modify some directory information. This is useful for checking information in the directory.

This section explains how to configure the web gateway to adapt it to the needs of your users, in particular to display new attributes or corporate profiles that you have defined.

Information on using the web gateway is provided in *Sun Directory Services 3.1 User's Guide*.

Configuring the Web Gateway

The gateway daemon, `dswebd`, requires the `dsservd` daemon to be running on the same machine. To enable users to browse the directory from any web browser, you must make sure that the `dsservd` and `dswebd` daemons are running. You can check their status in the Status section of the Admin Console. The LDAP service and the Web gateway service must be shown as *Running*. If they are not, use the Start button to start them.

You can change the default HTTP port (1760) used by the web gateway from the Admin Console. To do so, change the HTTP port number in the Web gateway section under Services.

You can modify the behavior of the web gateway and the way it displays information by editing the following configuration files:

- `dswebfilter.conf`

  Controls how the gateway makes a search request to the directory. See the `dswebfilter.conf(4)` man page for details.
- `dswebfriendly.conf`

  Contains user-friendly equivalents of certain attribute values that might be used in the directory. By default, it contains mappings between the ISO country codes and the names of countries.
- `dsweb.help`

  Contains help text for the user interface to the web gateway. You can edit this file and change the help text to reflect any changes you make to the user interface.
- `dsweb.helpattr`

  Contains an explanation of the directory attributes visible through the user interface to the web gateway.

- `dsweb.messages`

  Contains the messages and screen text used in the user interface to the web gateway. You can customize the user interface by changing this file.

- `dswebtmpl.conf`

  Contains templates that control how information retrieved from the directory is displayed. If you have modified the schema, especially if you have added object classes and attributes, you must modify this file to be able to display entries that use the new object classes and attributes. See the `dswebtmpl.conf(4)` man page for details.

Index