LDAP Setup and Configuration Guide

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94043-1100
U.S.A.

Part Number 806–5580–10
January, 2001

Contents

Tables

Figures

Preface

The *LDAP Setup and Configuration* minibook describes how to set up, configure and administer an LDAP client system. The information in this minibook will be incorporated into the *System Administration Guide: Naming Services* that is restructured to consolidate information from the *Solaris Naming Administration Guide* and *Solaris Naming Setup and Configuration Guide*.

Who Should Use This Book

The information in the *LDAP Setup and Configuration* minibook assumes that you are an experienced system and network administrator.

Although this manual introduces networking concepts relevant to LDAP as a Solaris name service, it does not explain LDAP concepts and networking fundamentals. It assumes that you are familiar with LDAP concepts, and have chosen your favorite administration tools.

Before You Read This Book

For information about Solaris name services, see the:

- *Solaris Naming Administration Guide*
- *Solaris Naming Setup and Configuration Guide*

If you are running iPlanet Directory Server 4.11, see the:

- iPlanet Directory Server installation instructions, Release Notes, and technical publications available at: `http://iPlanet.com`. iPlanet Directory Server 4.11 documents and Solaris Directory extension documents are also available on the *iPlanet Advantage Software, Volume I* CD.
- *Netscape Directory Server Schema Reference Guide*
- *Netscape Server Deployment Manual*
- *Managing Servers with Netscape Console 4.0*
- *Directory Server Administrator's Guide*

How This Book Is Organized

The *LDAP Setup and Configuration Guide* has the following organization:

Chapter 1 *Overview* introduces the LDAP model and briefly describes the LDAP operations.

Chapter 2 *Server Setup* provides background information about how to set up an LDAP directory server.

Chapter 4 *Client Setup* provides information about how to set up an LDAP client.

Chapter 3 *Netscape Directory Server Setup* provides an example scenario for configuring an iPlanet directory server to support Solaris LDAP Naming clients.

Appendix A *Schemas* describes the schemas required by LDAP to support Solaris LDAP Naming clients.

Appendix B *Troubleshooting the Configuration* briefly describes how to troubleshoot the configuration.

Related Books

For more information about deploying directory services see:

- Timothy A. Howes, Mark C. Smith, Gordon S. Good, *Understanding And Deploying LDAP Directory Services*, MacMillan Technical Publishing, 1999

Ordering Sun Documents

Fatbrain.com, an Internet professional bookstore, stocks select product documentation from Sun Microsystems, Inc.

For a list of documents and how to order them, visit the Sun Documentation Center on Fatbrain.com at `http://www1.fatbrain.com/documentation/sun`.

Accessing Sun Documentation Online

The docs.sun.com[SM] 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 Conventions 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

Overview

The *LDAP Setup and Configuration* guide describes how to set up an iPlanet LDAP directory server and how to set up a Solaris client to support the naming service.

Naming Service

Naming services store information in a central place that users, workstations, and applications must have to communicate across the networks. This information includes:

Machine (host) names and addresses
User names
Passwords
Group membership, and so on.

Without a central name service, each workstation would have to maintain its own copy of this information which makes it extremely expensive to administer large networks. Name service information can be stored in files, database tables and so on.

Solaris Name Services

The Solaris operating environment provides the following name services:

- DNS, the Domain Name System
- `/etc` files, the original UNIX naming system
- NIS, the Network Information Service
- NIS+, the Network Information Service Plus
- LDAP, the Lightweight Directory Access Protocol

For the detailed explanation of first four name services, refer to the *Solaris Naming Administration Guide.*

Most modern networks use a combination of two or more of these services that are coordinated by the name service switch, also known as the *switch.* The switch controls how a client workstation or application obtains network information. It determines which naming services an application uses to obtain naming information. For more information on Solaris switch, see `nsswitch.conf`(4).

LDAP Model

LDAP is the emerging industry standard protocol for accessing directory servers. It is a *lightweight* protocol. It is efficient, straight forward, and easy to implement, while still being highly functional. It uses a simplified set of system-independent encoding methods and runs directly on top of TCP/IP.

LDAP directories provide a way to name, manage, and access collections of directory entries. A directory *entry* is composed of attributes that have a type and one or more values. The syntax for each attribute defines the values allowed (such as ASCII characters or a jpeg photograph) and how those values are interpreted during a directory operation (such as whether a search or compare is case sensitive) .

Directory entries are organized into a tree structure, based on geographic (country), organizational (company) boundaries, or domains (dc).

Entries are named according to their position in this tree structure by a distinguished name (DN). Each component of the distinguished name is called a relative distinguished name (RDN). An RDN is composed of one or more attributes from the entry. (See RFC 2253 for a formal definition of a distinguished name.)

The hierarchy of the directory tree structure is analogous to that of the UNIX file system. An RDN is analogous to the name of a file, and the DN is analogous to the absolute pathname to the file. As in the UNIX file system, sibling directory entries must have unique RDNs. However, in the directory tree, both leaf nodes and nonleaf nodes can contain content or attributes.

Like the DNS namespace, LDAP directory entries are accessed in a “little-endian” manner. This means that LDAP names start with the least significant component and proceed to the most significant, that just below `root`. The DN is constructed by concatenating the sequence of RDNs up to the root of the tree. For example, if the person named Joe Qwerty works for the company named Ultra Keyboards in the United States, the commonName (CN) attribute for the person Joe Qwerty contains the value “Joseph Qwerty”. The DN contains “cn=Joseph Qwerty, o=Ultra Keyboards, c=US”.

Why LDAP as a Naming Service?

LDAP has the potential to replace existing application-specific directories and consolidate information. This means that changes made on an LDAP server will take effect for every directory-enabled application that uses this information. Imagine adding a variety of information about a new user through a single interface only once, and immediately the user has a Unix account, a mail address and aliases, membership in departmental mailing lists, access to a restricted Web server, and inclusion in job-specific restricted newsgroups. The user is also instantly included in the company’s phone list, mail address book, and meeting calendar system. When a user leaves, access can be disabled for all of these services with just a single operation.

A directory is distinguished from a general-purpose database by the usage pattern. A directory contains information that is often searched but rarely modified. Host names or user names, for example, are assigned once and then looked up thousands of times. LDAP servers are tuned for this type of usage, whereas relational databases are much more geared toward maintaining data that is constantly changing.

A directory can be replicated to protect from unfortunate situations like equipment failure by making the directory data available on multiple servers, known as replica servers. Replicas also improve performance by making more copies of directory data available and by placing the data close to the users and applications that use them.

Reducing load on the authoritative server is not the only reason for using replica servers. Many Unix networks use Network Information Service (NIS), also known as YP, which uses slave servers on each subnet. As with NIS, putting replicas on subnets can avoid network traffic through routers and reduce latency. However, unlike NIS, the LDAP synchronization scheme features incremental updates that can be pushed immediately to the replicas rather than periodically transferring all of the data.

In order for authoritative information to be maintained, access control needs to be imposed for privileges to read, write, search, or compare. Access control can be done on a subtree, entry, or attribute type and granted to individuals, groups, or "self" (which allows an authenticated user to access his or her own entry). This scheme provides a great deal of flexibility. For example, you may want to only allow people in a personnel department to change the title or manager attributes, allow

administrative assistants to change office location and pager number information for just their department, and allow individuals to modify their own home phone number, car license plate, and so on. For more information, check the iPlanet directory server documents.

Let's look at Unix login information as an example. Once attributes for users are stored in a directory server, you can synchronize user names and passwords for multiple operating system platforms when updated through Directory Server interface. This not only simplifies the change for users but can reduce the chance of having infrequently used accounts with forgotten passwords.

LDAP as a Naming Service in the Solaris Operating Environment

In Solaris, like NIS and NIS+, LDAP can also be used by the naming service switch to allow Solaris clients to obtain naming information.

The predominant protocol-independent interfaces to naming services within Solaris are the standard `getXbyY` APIs. An application using `getXbyY`() calls (e.g., `gethostbyname`(3NSL)) goes through the naming service switch which in turn calls the appropriate source protocol. In the case of LDAP, it calls LDAP APIs to retrieve information from a LDAP server. See `nsswitch.conf`(4) for more information about the naming service switch.

Figure 1–1 shows an overview of the relationship of the name services, the naming service switch, and the various parts of the LDAP implementation.

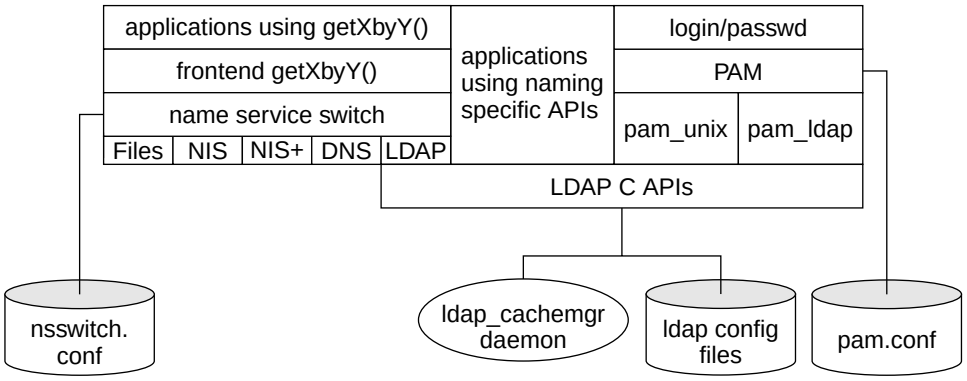

Figure 1–1 Architecture Overview

In addition to all the features of LDAP previously mentioned, the Solaris client configuration and maintenance is greatly simplified by storing client profiles in the directory. Each client runs a daemon that is responsible for refreshing the

configuration by downloading the latest profile from the directory. Once a change is required in client configuration (such as the addition of new LDAP servers, changes in security model, and so on), the system administration merely modifies the appropriate profile(s), and the clients will get the latest configuration automatically. See `ldap_cachemgr`(1M) for more information.

LDAP Operations

LDAP defines nine operation in three areas:

- Interrogation

  The `search` and `compare` operations interrogate the directory and retrieve its information.

- Update

  The `add`, `delete`, `modify`, and `modify RDN` operations update directory information.

- Authentication

  The `bind`, and `unbind` operations provide the groundwork for securing directory information. The `abandon` operation allows you to cancel an operation in progress.

CHAPTER 2

Server Setup

This chapter describes how to set up an LDAP server to support Solaris LDAP clients for naming information lookup. In particular, the setup allows Solaris LDAP clients to use the well-known `getXbyY` interfaces or `ldaplist`(1) to look up naming information on the LDAP server.

This chapter has the following organization:

Requirements

To support Solaris naming clients for naming information lookup the server must support the LDAP v3 protocol. This is necessary because Solaris Naming clients use controls that are available only in v3.

The following controls are available only in v3:

- Simple paged-mode (RFC 2696) .

- Virtual List View controls.

The server must support one of the following authentication methods:

- anonymous.
- SIMPLE (cleartext password).
- SASL CRAM-MD5.

▼ Verify that Directory Supports Simple Page Mode Control.

1. **Use ldapsearch to determine if the directory supports simple page mode control as identified by their OIDs: 1.2.840.113556.1.4.319 simple page mode control type and 2.16.840.1.113730.3.4.2 simple page mode control value.**

```
# ldapsearch -b "" -s base objectclass=\*
```

For our example configuration, `ldapsearch` returns:

```
objectclass=top
namingcontexts=dc=sun,dc=com,o=internet
subschemasubentry=cn=schema
supportedsaslmechanisms=CRAM-MD5
supportedextension=1.3.6.1.4.1.1466.20037
supportedcontrol=1.2.840.113556.1.4.319
supportedcontrol=2.16.840.1.113730.3.4.2
supportedldapversion=2
supportedldapversion=3
```

▼ Verify that Directory Supports Virtual List Views.

1. **Use ldapsearch to determine if the directory supports Virtual List Views as identified by their OIDs: 1.2.840.113556.1.4.473 VLV control type and 2.16.840.1.113730.3.4.9 VLV control value.**

```
# ldapsearch -b "" -s base objectclass=\*
```

For our example configuration, `ldapsearch` returns:

```
objectclass=top
namingcontexts=dc=sun,dc=com
namingcontexts=o=NetscapeRoot
subschemasubentry=cn=schema
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.4
supportedcontrol=2.16.840.1.113730.3.4.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=2.16.840.1.113730.3.4.9
supportedcontrol=2.16.840.1.113730.3.4.12
supportedsaslmechanisms=EXTERNAL
supportedldapversion=2
supportedldapversion=3
dataversion=atitrain2.east.sun.com:389 020000605172910
netscapemdsuffix=cn=ldap://:389,dc=atitrain2,dc=east,dc=sun,dc=com
```

Note - For more information on `ldapsearch` see `ldapsearch`(1).

Schemas

To support Solaris LDAP Naming clients, schemas defined by IETF and some Solaris specific schemas are required.

There are two required LDAP schemas defined by IETF: the RFC 2307 Network Information Service schema and the LDAP mailgroups Internet draft. To support Naming Information Service, the definition of these schemas must be added to the directory server. Detailed information about IETF and Solaris specific schemas is included in Appendix A. The various RFCs can also be accessed on web at IETF site `http://www.ietf.org`.

Directory Information Tree

Solaris LDAP clients use the information in a default Directory Information Tree (DIT) . This default DIT, however, can be overridden by specifying the modified DIT

in the profile. The DIT is divided into containers that are subtrees containing entries for a specific information type.

The search `baseDN` specifies the location in the DIT where all information for the client is found. In the node designated as the search base, the `NisDomainObject` objectclass must exist. The search base node subtrees designate all the containers for the various information types. See Figure 2–1.

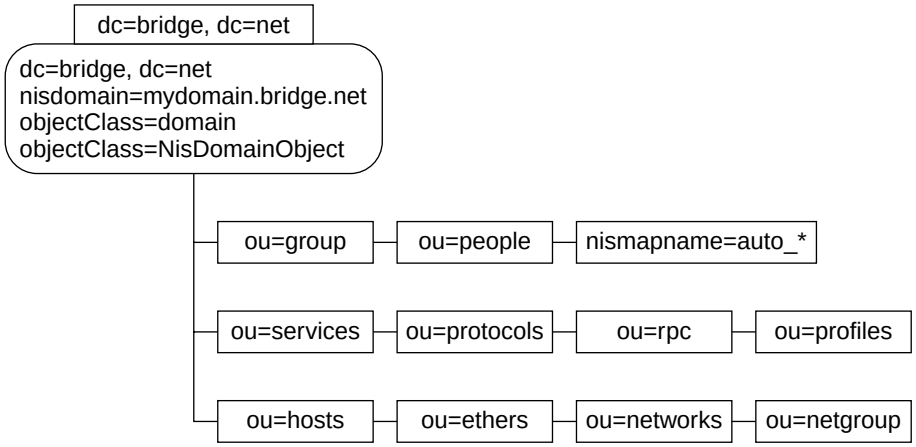

Figure 2–1 Directory Information Tree Containers

Table 2–1 lists the container and information type stored in the DIT:

TABLE 2–1 Directory Information Tree

Container	Information Type
ou=Ethers	bootparams(4), ethers(4)
ou=Group	group(4)
ou=Hosts	hosts(4), ipnodes(4),publickey(4)
ou=Aliases	aliases(4)
ou=Netgroup	netgroup(4)
ou=Networks	networks(4), netmasks(4)
ou=People	passwd(1), shadow(4), user_attr(4),audit_user(4), publickey for users
ou=Protocols	protocols(4)
ou=Rpc	rpc(4)
ou=Services	services(4)
ou=SolarisAuthAttr	auth_attr(4)

TABLE 2–1 Directory Information Tree *(continued)*

Container	Information Type
ou=SolarisProfAttr	prof_attr(4), exec_attr(4)
ou=projects	project
nismapname=auto_*	auto_*

Override the Default Containers in the DIT

If a particular LDAP deployment requires the default containers be overridden, it is possible to do so by specifying the modified container in the profile. You can define an alternate search `baseDN` for each of the databases

For example, assume that an organization wants to replace the `ou=People` container with `ou=employee` and `ou=contractor` containers. For this profile entry (which can exist anywhere in the DIT), an alternate search DN needs to be specified. Generate the LDAP client profile using the `-B` option to specify an alternate search DN. See `ldap_gen_profile`(1M) for details. The attribute looks like:

```
SolarisDataSearchDN="passwd:(ou=employee,dc=mkt,dc=mystore,dc=com),
                     (ou=contractor,dc=mkt,dc=mystore,dc=com)"
```

NIS Domain

In order for the Solaris clients to find a server for a specific domain, the `nisDomain` attribute of the `nisDomainObject` objectclass must be defined in the root DN entry of the DIT representing the desired domain. This information is used by the client when initializing the system and refreshing the client profile. During the initialization, the client searches for an entry on the LDAP server that has the `nisDomain` matching the desired domain. The DN for the entry found will be used as the `BaseDN` for the naming information.

When refreshing the client profile, the `ldap_cachemgr` on the client machine verifies that `nisDomain` defined in the root DN entry matches the domain desired before refreshing its profile.

For illustrative purposes, this document uses the following `nisDomain`:

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
objectclass: top
objectclass: domain
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
```

Client Profile

To simplify Solaris client setup, a client profile needs to be defined. This profile must be created on the server. During the initialization stage, a client can easily set up the system with the profile name and the server's address. The client profile allows the system administrators to define the LDAP environment to be used by Solaris clients.

The most obvious benefit of using a profile, is the ease of installing a machine. However, the true benefit of using profiles only becomes apparent when you start making changes in your environment (such as adding or removing servers). See `ldap_gen_profile`(1M) for details.

The following list shows the possible attributes that can be defined in the profile;

- `SolarisLDAPServers`

  a comma separated list of LDAP server ip addresses with the optional colon separated port numbers that can be used by the client. There is no default for this parameter, and at least one LDAP server must be defined. In case of multiple servers, if the first server response is not retrieved, the next server is contacted.

- `SolarisSearchBaseDN`

  the LDAP Naming base DN where the Naming information will be stored.

- `SolarisBindDN`

  the LDAP identity used during the authentication process by the clients. Usually this is the proxy agent DN. The default is a `NULL` string.

- `SolarisBindPassword`

  the password of `SolarisBindDN` when using `SIMPLE` and `CRAM_MD5` authentication. The default is a `NULL` string.

- `SolarisAuthMethod`

  the ordered list of authentication method(s) to be used by the clients. Possible methods include: `NONE`, `SIMPLE` or `CRAM_MD5`. The default is `NONE`. In case of multiple methods, if the first authentication method does not succeed, (except due to credentials), the next one is tried.

- `SolarisTransportSecurity`

  the secured transport to be used by the client. The default is `NONE`; currently `NONE` is the only option supported.

- `SolarisDataSearchDN`

  alternate `baseDN` when searching for Naming information. This allows you to override the default naming information type. The alternate `baseDN` consists of following format:

  ```
  database:alternate-baseDN-list
  ```

  The `database` is the information type defined in the `nsswitch.conf` file, and the `alternate-baseDN-list` is a list of alternate baseDNs enclosed with parentheses and separated by a comma. The lookup to a specific database is done in the order specified in this parameter. The default for all containers is `NULL`.

- `SolarisSearchScope`

  search scope to be used when looking up Naming information. Possible values are: `Base`, `One level`, or `Subtree`. Default is `One level`.

- `SolarisSearchTimeLimit`

  LDAP search time limit in seconds when searching for Naming information. The default is 30 seconds.

- `SolarisCacheTTL`

  Time-To-Live value for clients to refresh their profile information from the server. Set `client_TTL` to `0` (zero) if you do not want `ldap_cachemgr` to attempt an automatic refresh from the servers. The times are specified with either a zero `0` (for no expiration) or a positive integer in number of seconds. The default is 43200 (which is 12h).

- `SolarisSearchReferral`

  referral option to be used when looking up Naming information. Possible options are follow referral, or do not follow referral. Default is to always follow referrals.

The `ldap_gen_profile`(1M) command is provided as part of the Solaris client tools to create client profiles. This tool generates an `LDIF` file which can be stored in the LDAP server using the `ldapadd`(1) command. The following example shows how to create a client profile.

▼ How to Create a Client Profile

1. Use `ldap_gen_profile`(1M) to create a client profile.

```
# /usr/sbin/ldap_gen_profile \
-P myprofile \
```

(continued)

(Continuation)

```
-b dc=mkt,dc=mainstore,dc=com \
-a simple -w mypasswd \
-D cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com \
100.100.100.100
```

The following example shows the profile generated:

```
dn: cn=myprofile,ou=profile,dc=mkt,dc=mainstore,dc=com
SolarisBindDN: cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com
SolarisBindPassword: {NS1}xxxxxxxxxxxxx
SolarisLDAPServers: 100.100.100.100
SolarisSearchBaseDN: dc=mkt,dc=mainstore,dc=com
SolarisAuthMethod: NS_LDAP_AUTH_SIMPLE
SolarisTransportSecurity: NS_LDAP_SEC_NONE
SolarisSearchReferral: NS_LDAP_FOLLOWREF
SolarisSearchScope: NS_LDAP_SCOPE_ONELEVEL
SolarisSearchTimeLimit: 30
SolarisCacheTTL: 43200
cn: myprofile
ObjectClass: top
ObjectClass: SolarisNamingProfile
```

2. Save the generated profile to a file (such as profile.ldif) and use `ldapadd`(1) to store the client profile file in the LDAP server.

```
# ldapadd -h ldap_server_hostname -D "cn=Directory Manager" \
-w nssecret -f profile.ldif
```

The `ldap_cachemgr`(1M) on every client machine automatically updates the content of the LDAP configuration files. This means changes need to be made only on the server and those changes automatically propagate to every client in the namespace. The periodic update is based on the TTL, time to live value specified in the profile `SolarisCacheTTL`.

Security Model

To access the information stored in the directory, clients must authenticate to the directory first. Once authenticated, and depending on the authorization information stored in the directory in the form of Access Control Information the client will have access to part or all of the information available in the directory. In this section, the concepts of client identity, authentication methods, and finally PAM modules will be discussed. For more information on ACI, please consult the iPlanet Directory Server Administrator's Guide.

Authentication Identity

LDAP name services could be configured to use one of two possible identities for authenticating clients to the directory: anonymous, and proxy agent.

- Anonymous

  Authentication is all about establishing identity, and anonymous is considered a special case of identity. Obviously anonymous does not provide any level of security and means that all unauthenticated connections to the directory will be able to browse and read all Network Information records (including password and shadow information). However even with the absence of security it might be an appropriate choice for some installations and it is allowed.

- Proxy Agent

  In case of proxy agent identity, the client authenticates to the directory using a proxy account in the directory. This proxy account can be any entry that is allowed to bind to the directory (in the iPlanet Directory Server, this translates to any entry which has a `userPassword` attribute).

  Access control to parts of the information in the directory can be achieved by setting appropriate ACI's restricting or granting various rights based on the proxy's identity. Furthermore, since there is no relationship between the number of proxy agents and clients, you can have any combination of the two. For example, in one extreme a deployment could have one proxy agent for all its clients and grant the proxy read access to all parts of the DIT that contain naming information. On the other hand one could setup a server where each client authenticates using a different proxy agent and can set the ACI to restrict access per client. These examples demonstrate two extreme cases of using proxy agents; however, a typical deployment lies somewhere in between the two extremes. The granularity level of this is a choice for the directory architect, and must be considered carefully. Too few proxy agents might limit the ability of the system administrator to control access to resources, but too many agents complicates the setup and maintenance of the system as it would require a large number of profiles as well.

Note - Because client configuration is stored in profiles, there is a direct relationship between the number of proxies used and profiles that need to be defined.

Authentication Method

When a proxy agent is used the system administrator also needs to choose an authentication method for that identity to authenticate to the directory. Currently the supported mechanisms by Solaris 8 clients are `SIMPLE`, and `CRAM-MD5`.

- `SIMPLE`

  If `SIMPLE` is chosen, the client authenticates to an LDAP server by sending a simple bind request to the server. It is worth noting that with this authentication method, the password is transmitted in the clear and is subject to snooping. The primary advantage of using `SIMPLE` is that it is the required authentication method as defined in the LDAP standard, and all directory servers support it.

- `CRAM-MD5`

  Some directory servers also support Challenge Response Authentication Mechanism (`CRAM-MD5`) through Simple Authentication and Security Layer (`SASL`). The primary advantage of `CRAM-MD5` is that the password does not go over the wire in the clear during authentication and therefore is more secure than `SIMPLE`. See RFC 2195 for information on `CRAM-MD5`. See RFC 2222 for information on `SASL`.

Note - Currently the iPlanet Directory Server version 4.11 does not support the `CRAM-MD5` method.

Pluggable Authentication Module (PAM)

PAM provides a way for applications to remain independent of authentication scheme used in the Solaris Operating Environment. By using the PAM layer, applications can perform authentication without worrying about what authentication method is defined by the system administrator for the given client. To use LDAP naming service, one of two pam modules can be configured in `pam.conf`: `pam_unix`(5) and `pam_ldap`(5).

- `pam_unix`

  When `pam_unix` is used, the traditional model of UNIX authentication is followed which means that the encrypted password of the user is retrieved from the directory to the local machine, the user is prompted for his password, user's password is encrypted, and finally the two encrypted passwords are compared to decide if the user should be authenticated or not. If clients using LDAP are

configured with this module, the `userPassword` attribute must be readable by the identity that the client is using (anonymous or the configured proxy agent). Additionally, there are two more restrictions when using `pam_unix`:

1. The password must be stored in an attribute called `userPassword`.
2. The password must be stored in `UNIX` `crypt` format (not clear text or encrypted by other encryption methods).

- `pam_ldap`

  Since the traditional method of authentication used by `pam_unix` is not necessarily the best option when deploying LDAP directories, a new PAM module was added in Solaris 8 which authenticates users directly to the directory instead. This will allow Solaris clients to work with newer and more advanced authentication methods that the directory server might support. By definition, clients using `pam_ldap` do not require read access to the password attribute, and they do not need the password to be stored in any specific format in the directory.

  As an added benefit, because `pam_ldap` authenticates users directly to the directory server, user level access controls can be put in place to control an individuals' authentication using ACI's.

  As with using `pam_unix`, use the `passwd` command to change a password.

Indexes

Most LDAP servers use indexes to improve the search performance. To use indexes, consult your directory server documentation.

In addition to the basic indexed attributes, the following list of attributes must be indexed to ensure that Solaris clients will be able to retrieve naming information in a reasonable time.

```
membernisnetgroup    pres,eq,sub
nisnetgrouptriple    pres,eq,sub
memberuid            pres,eq
macAddress           pres,eq
uid                  pres,eq
uidNumber            pres,eq
gidNumber            pres,eq
ipHostNumber         pres,eq
ipNetworkNumber      pres,eq
ipProtocolNumber     pres,eq
oncRpcNumber         pres,eq
ipServiceProtocol    pres,eq
ipServicePort        pres,eq
nisDomain            pres,eq
```

(continued)

```
nisMapName          pres,eq
mail                pres,eq
```

Note - The abbreviations used in attribute list expand to: `pres` is presence, `eq` is equality, and `sub` is substring

Additionally, if Virtual List View control (vlv) is supported by the server, vlv indexes also need to be created. Create these indexes for any container in the tree that contains a large number of objects. (Large is relative to other objects in the tree.) For information on how to create these indexes, consult your directory server documentation. Ensure that the vlv sort value is set to `cn uid` and the vlv filter and scope are defined per the following list:

```
getpwent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getspent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getgrent:      vlvFilter: (objectclass=posixGroup),       vlvScope: 1
gethostent:    vlvFilter: (objectclass=ipHost),           vlvScope: 1
getnetent:     vlvFilter: (objectclass=ipNetwork),        vlvScope: 1
getprotoent:   vlvFilter: (objectclass=ipProtocol),       vlvScope: 1
getrpcent:     vlvFilter: (objectclass=oncRpc),           vlvScope: 1
getaliasent:   vlvFilter: (objectclass=rfc822MailGroup),  vlvScope: 1
getserviceent: vlvFilter: (objectclass=ipService),        vlvScope: 1
```

The Cost of Indexing

Although indexes improve search performance, they incur the following costs:

- Slower database modification.

  The more indexes you maintain, the longer it takes to update the database. This is especially true for substring indexes that cause the directory server to generate multiple index files whenever an attribute value is created or changed. For substring indexes, the number of index entries created is proportional to the length of the string being indexed.

- Additional system resources required.

  - Additional disk space

    The more attributes you index, the more files the directory server creates.

  - Additional memory.

    To run more efficiently, the directory server maintatins as many index files as possible in memory; thus, making a greater demand for memory space.

  - Increased disk activity.

Maintaining indexes that are not frequently accessed creates indexes that might be minimally used and swapped to disk when more frequently accessed index files are paged from disk.

Loading Data

Solaris Directory extension package has tools and documents needed to migrate from NIS to LDAP. It is available on *iPlanet Advantage Software (Volume 1)* cd, and also on iPlanet website. `dsimport` is a tool used to convert NIS data to ldap format. It makes use of a mapping file, `nis.mapping`, which needs to be customized as per the input data format, environment variables and so on. For detailed information on these, refer to above mentioned cd or website.

Command Line Tools

LDAP provides command line tools that correspond to the operations performed by the LDAP API. Each tool supports a common set of options, including authentication and bind parameters.

- `ldapsearch`

  Search for directory entry. Display attributes and values found.

- `ldapmodify`

  Modify, add, delete, or rename directory entry.

- `ldapadd`

  Add new directory entry.

- `ldapdelete`

  Delete existing directory entry.

- `ldapmodrdn`

  Rename existing directory entry.

The `ldapsearch`, `ldapadd`, and `ldapmodify` tools support a common text-based format for representing directory information called the LDAP Data Interchange Format (LDIF).

LDAP Data Interchange Format

LDIF is the format produced by the `ldapsearch` tool, the format accepted by the `ldapadd` tool, and is the basis for the change information format that the `ldapmodify` tool uses.

An LDIF file contains one or more entries. Each entry is separated by an empty line. The basic form on an LDIF file entry is:

```
 [id]
dn: entryDN
attrtype: attrvalue
...
```

where:

- `id`

  is an optional numeric entry identifier (not used by the LDAP tools).

- `entryDN`

  is the LDAP Distinguished Name (DN) of the directory entry.

- `attrtype`

  is an LDAP attribute type, such as `cn` or `telephoneNumber`.

- `attrvalue`

  is a value for `attrtype`.

The `attrtype`: `attrvalue` line can be repeated as many times as necessary to list all of the attribute values present in an entry. The line can be continued by inserting a single space or horizontal tab character at the start of the next line.

For example, an LDIF file that contains Joe Qwerty's entry includes five attributes (`cn` and `objectclass` have two values):

```
dn: cn=Joseph Qwerty, o=Ultra Keyboards Inc., c=US
cn: Joseph Qwerty
cn: Joe Qwerty
sn: Qwerty
mail: jqwerty@ultra.com
seeAlso: cn=Joe Qwerty, ou=Engineering Division, o=Peo
 ple, o=IEEE, c=US
objectClass: top
objectClass: person
```

Note - The value of `seeAlso` is split across two lines by inserting a single space character at the start of the line that begins with " `ple, ...`".

▼ How to Search the Directory

Use `ldapsearch`(1) to find a directory entry. `ldapsearch` opens a connection to the LDAP directory server, binds to the directory server, and performs a search of the directory.

1. Find members of IEEE that work at Ultra Keyboards in the United States.

```
% ldapsearch -L -b "o=IEEE, o=Ultra Keyboards Inc., c=US" uid=\*
```

The results of the search are:

```
dn: uid=jqwerty, o=IEEE, o=Ultra keyboards Inc., c=US
uid: jqwerty
cn: jqwerty
userpassword: {crypt}somecryptedtext
uidnumber: 12345
gidnumber: 123
gecos: Joseph Qwerty
homedirectory: /home/jqwerty
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
shadowlastchange: 3455

dn: uid=bhand, o=IEEE, o=Ultra keyboards Inc., c=US
uid: bhand
cn: bhand
userpassword: {crypt}somecryptedtext
uidnumber: 12347
gidnumber: 123
gecos: William Handset
homedirectory: /home/bhand
loginshell: /bin/csh
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
shadowlastchange: 3440
```

▼ How to Modify a Directory Entry

Use `ldapmodify`(1) to change a directory entry. `ldapmodify` opens a connection to the LDAP directory server, binds to the directory server, and performs a sequence of LDAP modify operations on the directory.

1. Bind as the directory manager (password “enigma”) and add email address eng@ultra.com Joe Qwerty entry

```
% ldapmodify -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma < modfile
```

The contents of `modfile` are:

```
dn: cn=carol,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}mgq25KV6CE0p6
-
replace: objectclass
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 6447
-

dn: cn=stephen,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}w.4P1JPV3w.Zs
-
replace: objectclass
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 6447
-

dn: cn=frank,ou=People,o=Ultra Keyboards Inc.,c=US
changetype: modify
replace: userpassword
userpassword: {crypt}mMBEaHRlf5rJQ
```

(continued)

```
-
replace: objectclass
objectclass: top
objectclass: shadowAccount
objectclass: account
objectclass: posixAccount
-
add: shadowlastchange
shadowlastchange: 9712
-
```

Note - A line with just a hyphen separates a series of modification commands for the same directory entry. A blank lines separates different directory entries.

If the operation is successful, `ldapmodify` returns a message similar to the following:

```
# ldapmodify -D "cn=Directory Manager" -w nssecret -f domain.ldif
modifying entry dc=sun,dc=com
```

If unsuccessful an error message is displayed.

▼ How to Add an entry to the Directory

Use `ldapadd`(1) to add an entry to the directory. `ldapadd` opens a connection to the LDAP directory server, binds to the directory server, and performs a sequence of LDAP add operations on the directory.

1. **Bind as the directory manager (password “enigma”) and add an entries for Penny Gold and Amy Lamb.**

```
% ldapadd -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma < addfile
```

The contents of `addfile` are:

```
dn: cn=Penny Gold, o=Ultra Keyboards Inc., c=US
changetype: add
objectclass: top
objectclass: person
objectclass: inetOrgPerson
cn: Penny Gold
sn: Gold
mail: pgold@ultra.com

dn: cn=Amy Lamb, o=Ultra Keyboards Inc., c=US
changetype: add
objectclass: top
objectclass: person
objectclass: inetOrgPerson
cn: Amy Lamb
sn: Lamb
mail: alamb@ultra.com
```

▼ How to Delete an entry From the Directory

Use `ldapdelete`(1) to delete an entry from the directory. `ldapdelete` opens a connection to the LDAP directory server, binds to the directory server, and performs one or more LDAP delete entry operations on the directory.

1. **Bind as the directory manager (password “enigma”) and delete the entry for Penny Gold.**

```
% ldapdelete -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma "cn=Penny Gold, o=Ultra Keyboards Inc., c=US"
```

`ldapdelete` returns nothing if the operation is successful; otherwise, an error message is displayed.

▼ How to Rename a Directory Entry

Use `ldapmodrdn`(1) to rename an existing directory entry. `ldapmodrdn` opens a connection to the LDAP directory server, binds to the directory server, and performs one or more LDAP modify RDN (rename) operations on the directory.

1. **Bind as the directory manager (password “enigma”) and change the RDN cn value from “User Interface” to “Ergonomic”.**

```
% ldapmodrdn -r -D "cn=Manager, o=Ultra Keyboards Inc., \
c=US" -w enigma "cn=User Interface, o=Ultra Keyboards Inc., \
c=US" "cn=Ergonomic"
```

`ldapmodrdn` returns nothing if the operation is successful; otherwise, an error message is displayed.

CHAPTER 3

iPlanet Directory Server Setup

This chapter describes how to set up the iPlanet Directory Server to support Solaris LDAP clients for Naming information lookup. The information is specific to version 4.11 of the iPlanet Directory Server.

If you are using the iPlanet Directory Server, see the following iPlanet documents:

- *Netscape Directory Server Schema Reference Guide*
- *Netscape Server Deployment Manual*
- *Managing Servers with Netscape Console 4.0*
- *Directory Server Administrator's Guide*

This chapter has the following organization:

Add Object Class Definitions to the Configuration Directory

▼ Prepare the Environment.

1. Stop the directory server.

▼ Modify the slapd.oc.conf File.

1. **Modify the ipNetwork object class so cn is no longer required, but is still a member.**

   `ipNetwork` before the change:

```
objectclass ipNetwork
    oid
         1.3.6.1.1.1.2.7
    requires
         objectClass,
         ipNetworkNumber,
         cn
    allows
         ipNetmaskNumber,
         manager,
         l,
         description
```

   Remove the `cn` line from `requires`; add the `cn` line to `allows`. `ipNetwork` after the change:

```
objectclass ipNetwork
    oid
         1.3.6.1.1.1.2.7
    requires
         objectClass,
         ipNetworkNumber
    allows
         cn,
         ipNetmaskNumber,
         manager,
         l,
         description
```

▼ Add Object Class Definitions to the slapd.user_oc.conf File

1. **Add the NisKeyObject objectclass.**

```
# NIS publickey objectclass
objectclass NisKeyObject
        oid 1.3.6.1.1.1.2.14
        superior top
        requires
                cn,
                nisPublickey,
                nisSecretkey
        allows
                uidNumber,
                description
```

2. Add the nisDomainObject objectclass.

```
# NIS domain objectclass
objectclass nisDomainObject
        oid 1.3.1.6.1.1.1.2.15
        superior top
        requires
                nisDomain
```

3. Add the SolarisNamingProfile objectclass.

```
# LDAP client profile objectclass
objectclass SolarisNamingProfile
        oid 1.3.6.1.4.1.42.2.27.5.2.7
        superior top
        requires
                cn,
                SolarisLDAPservers,
                SolarisSearchBaseDN
        allows
                SolarisBindDN,
                SolarisBindPassword,
                SolarisAuthMethod,
                SolarisTransportSecurity,
                SolarisCertificatePath,
                SolarisDataSearchDN,
                SolarisSearchScope,
                SolarisSearchTimelimit,
                SolarisPreferredServer,
                SolarisPreferredServerOnly,
                SolarisCacheTTL,
```

(continued)

```
                SolarisSearchReferral
```

4. Add the mailGroup objectclass.

```
# mailGroup objectclass
objectclass mailGroup
        oid 2.16.840.1.113730.3.2.4
        superior top
        requires
                mail
        allows
                cn,
                mgrpRFC822MailMember
```

5. Add the nisMailAlias objectclass.

```
# nisMailAlias objectclass
objectClass nisMailAlias
        oid 1.3.6.1.4.1.42.2.27.1.2.5
        superior top
        requires
                cn
        allows
                rfc822mailMember
```

6. Add the nisNetId objectclass.

```
# nisNetId objectclass
objectClass nisNetId
        oid 1.3.6.1.4.1.42.2.27.1.2.6
        superior top
        requires
                cn
        allows
                nisNetIdUser,
                nisNetIdGroup,
                nisNetIdHost
```

7. Add the SolarisAuditUser objectclass.

```
# User auditing objectclass
objectclass SolarisAuditUser
        oid 1.3.6.1.4.1.42.2.27.5.2.2
        superior top
        allows
                SolarisAuditAlways,
                SolarisAuditNever
```

8. Add the SolarisUserAttr objectclass.

```
# RBAC User attributes objectclass
objectclass SolarisUserAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.3
        superior top
        allows
                SolarisUserQualifier,
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrKeyValue
```

9. Add the SolarisAuthAttr objectclass.

```
# RBAC Authorizations Objectclass
objectclass SolarisAuthAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.4
        superior top
        requires
                cn
        allows
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrShortDesc,
                SolarisAttrLongDesc,
                SolarisAttrKeyValue
```

10. Add the SolarisProfAttr objectclass.

```
# RBAC Profile objectclass
objectClass SolarisProfAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.5
        superior top
        requires
                cn
        allows
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisAttrLongDesc,
                SolarisAttrKeyValue
```

11. Add the SolarisExecAttr objectclass.

```
# RBAC Execution objectlcass
objectClass SolarisExecAttr
        oid 1.3.6.1.4.1.42.2.27.5.2.6
        superior top
        allows
                SolarisKernelSecurityPolicy,
                SolarisProfileType,
                SolarisAttrReserved1,
                SolarisAttrReserved2,
                SolarisProfileID,
                SolarisAttrKeyValue
```

12. Add the nisKeyObject objectclass.

```
# Publickey objectclass
objectClass nisKeyObject
        oid 1.3.6.1.1.1.2.14
        superior top
        requires
                cn,
                nisPublicKey,
                nisSecretKey
        allows
                uidNumber,
                description
```

13. Add the SolarisProject objectclass.

```
# Project Accounting objectclass
objectclass SolarisProject
        oid 1.3.6.1.4.1.42.2.27.5.2.1
        superior top
        requires
                SolarisProjectID,
                SolarisProjectName
        allows
                memberUid,
                memberGid,
                description,
                SolarisProjectAttr
```

▼ Add Attribute Definitions to the slapd.user_at.conf File

1. Add the nisMapEntry attribute.

```
# Sun nisMapEntry attributes
attribute nisDomain      1.3.6.1.1.1.1.30   cis
```

2. Add the LDAP client profile attributes.

```
# attributes for LDAP client profile
attribute SolarisLDAPServers     1.3.6.1.4.1.42.2.27.5.1.15   cis
attribute SolarisSearchBaseDN    1.3.6.1.4.1.42.2.27.5.1.16   dn single
attribute SolarisCacheTTL        1.3.6.1.4.1.42.2.27.5.1.17   cis  single
attribute SolarisBindDN          1.3.6.1.4.1.42.2.27.5.1.18   dn single
attribute SolarisBindPassword    1.3.6.1.4.1.42.2.27.5.1.19   ces  single
attribute SolarisAuthMethod      1.3.6.1.4.1.42.2.27.5.1.20   cis
attribute SolarisTransportSecurity  1.3.6.1.4.1.42.2.27.5.1.21   cis
attribute SolarisCertificatePath 1.3.6.1.4.1.42.2.27.5.1.22   ces single
attribute SolarisDataSearchDN    1.3.6.1.4.1.42.2.27.5.1.24   cis
attribute SolarisSearchScope     1.3.6.1.4.1.42.2.27.5.1.25   cis single
attribute SolarisSearchTimeLimit 1.3.6.1.4.1.42.2.27.5.1.26   int single
attribute SolarisPreferredServer 1.3.6.1.4.1.42.2.27.5.1.27   cis
attribute SolarisPreferredServerOnly 1.3.6.1.4.1.42.2.27.5.1.28 cis single
attribute SolarisSearchReferral  1.3.6.1.4.1.42.2.27.5.1.29   cis single
```

3. Add the mailGroup attributes.

```
# Sun additional attributes to RFC2307 attributes (NIS)
attribute mgrpRFC822MailMember   2.16.840.1.113730.3.1.30     cis
attribute rfc822MailMember                                    ces
```

4. Add the nisKeyObject attributes.

```
# Sun nisKeyObject attributes
attribute nisPublickey    1.3.6.1.1.1.1.28    cis
attribute nisSecretkey    1.3.6.1.1.1.1.29    cis
```

5. Add the nisNetId attributes.

```
# Sun nisNetId attributes
attribute nisNetIdUser    1.3.6.1.4.1.42.2.27.1.1.12    ces
attribute nisNetIdGroup   1.3.6.1.4.1.42.2.27.1.1.13    ces
attribute nisNetIdHost    1.3.6.1.4.1.42.2.27.1.1.14    ces
```

6. Add the auditing attributes.

```
# attributes for auditing
attribute SolarisAuditAlways   1.3.6.1.4.1.42.2.27.5.1.5   cis single
attribute SolarisAuditNever    1.3.6.1.4.1.42.2.27.5.1.6   cis single
```

7. Add the RBAC attributes.

```
# attributes for RBAC
attribute SolarisAttrKeyValue   1.3.6.1.4.1.42.2.27.5.1.4   cis single
attribute SolarisAttrShortDesc  1.3.6.1.4.1.42.2.27.5.1.7   cis single
attribute SolarisAttrLongDesc   1.3.6.1.4.1.42.2.27.5.1.8   cis single
attribute SolarisKernelSecurityPolicy  1.3.6.1.4.1.42.2.27.5.1.9
   cis single
attribute SolarisProfileType    1.3.6.1.4.1.42.2.27.5.1.10  cis single
attribute SolarisProfileId      1.3.6.1.4.1.42.2.27.5.1.11  ces single
attribute SolarisUserQualifier  1.3.6.1.4.1.42.2.27.5.1.12  cis single
attribute SolarisAttrReserved1  1.3.6.1.4.1.42.2.27.5.1.13  cis single
attribute SolarisAttrReserved2  1.3.6.1.4.1.42.2.27.5.1.14  cis single
```

8. Add the nisKeyObject attributes.

```
# attributes for nisKeyObject
attribute nisPublicKey    1.3.6.1.1.1.1.28    cis
attribute nisSecretKey    1.3.6.1.1.1.1.29    cis
```

9. Add the project accounting attributes.

```
# attributes for Project Accounting
attribute SolarisProjectID     1.3.6.1.4.1.42.2.27.5.1.1    int single
attribute SolarisProjectName   1.3.6.1.4.1.42.2.27.5.1.2    ces single
attribute SolarisProjectAttr   1.3.6.1.4.1.42.2.27.5.1.3    ces
attribute memberGid            1.3.6.1.4.1.42.2.27.5.1.30   ces
```

Load Data Into the Directory Server

If not already configured, configure directory server to store passwords in `Unix Crypt` format. For more information on setting the password `Unix Crypt` format, see the iPlanet documents.

▼ Set the ACI

1. **Set the ACI for the top entry of your tree. This ACI controls an owners ability to modify their own entry. For instance, the default ACI allows a user to modify their home directory value. The modified ACI does not. You might need to set the ACI specific to your environment.**

   Change the "Allow self entry modification" ACI of the top entry of your tree from:

```
aci=(targetattr = "*")(version 3.0; acl "Allow self entry modification";
allow (write)userdn = "ldap:///self";)
```

   The modified ACI is:

```
aci=(targetattr!="cn || uid || uidNumber || gidNumber || homeDirectory
|| shadowLastChange || shadowMin || shadowMax || shadowWarning ||
shadowInactive || shadowExpire || shadowFlag || memberUid")
(version 3.0; acl "Allow self entry modification"; allow
(write) userdn = "ldap:///self"; )
```

Note - Do not give modify permission for attributes which the user should not be able to change, such as `uid`. Doing so allows the user to become super user by setting the attribute to 0.

▼ Add the Naming Container Entries.

For a list of naming containers, see “Directory Information Tree” on page 23.

Note - The following container entries are based on the `nisDomain` example used in “NIS Domain” on page 25. Change the container entries as they apply to your environment.

---

1. **Add the domain entry.**

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
associatedDomain: mkt.mainstore.com
objectClass: top
objectClass: domain
objectClass: domainRelatedObject
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
```

2. Add the naming container entries.

```
dn: ou=people,dc=mkt,dc=mainstore,dc=com
ou: people
objectClass: top
objectClass: organizationalUnit

dn: ou=Group,dc=mkt,dc=mainstore,dc=com
ou: Group
objectClass: top
objectClass: organizationalUnit

dn: ou=rpc,dc=mkt,dc=mainstore,dc=com
ou: rpc
objectClass: top
objectClass: organizationalUnit

dn: ou=protocols,dc=mkt,dc=mainstore,dc=com
ou: protocols
objectClass: top
objectClass: organizationalUnit

dn: ou=networks,dc=mkt,dc=mainstore,dc=com
ou: networks
objectClass: top
objectClass: organizationalUnit

dn: ou=netgroup,dc=mkt,dc=mainstore,dc=com
ou: netgroup
objectClass: top
objectClass: organizationalUnit

dn: ou=aliases,dc=mkt,dc=mainstore,dc=com
ou: aliases
objectClass: top
objectClass: organizationalUnit

dn: ou=Hosts,dc=mkt,dc=mainstore,dc=com
ou: Hosts
objectClass: top
objectClass: organizationalUnit
```

(continued)

(Continuation)

```
dn: ou=services,dc=mkt,dc=mainstore,dc=com
ou: services
objectClass: top
objectClass: organizationalUnit

dn: ou=Ethers,dc=mkt,dc=mainstore,dc=com
ou: Ethers
objectClass: top
objectClass: organizationalUnit

dn: ou=profile,dc=mkt,dc=mainstore,dc=com
ou: profile
objectClass: top
objectClass: organizationalUnit

dn: nismapname=auto_home,dc=mkt,dc=mainstore,dc=com
nismapname: auto_home
objectClass: top
  objectClass: nisMap

dn: nismapname=auto_direct,dc=mkt,dc=mainstore,dc=com
nismapname: auto_direct
objectClass: top
objectClass: nisMap

dn: nismapname=auto_master,dc=mkt,dc=mainstore,dc=com
nismapname: auto_master
objectClass: top
objectClass: nisMap

dn: nismapname=auto_shared,dc=mkt,dc=mainstore,dc=com
nismapname: auto_shared
objectClass: top
objectClass: nisMap
```

▼ Set Performance and Limit Parameters

The value of each of these parameter varies from server to server; such as how much data is loaded, usage pattern, the available hardware and so forth.

1. **Set the following performance parameters: Maximum entries in cache, Maximum cache size (bytes), and look through limit.**

   Modify the caching parameters to accomodate the memory and disk space available on your system.

2. **Set the following limits parameters: size and time limit for your environment.**

   Specifying `sizelimit` or `timelimit` to `-1`, sets them to their maximum value. Select values to accomodate your system.

▼ Give the Proxy Agent Read Permission for Password

Note - The following proxy agent ACI information is based on the `nisDomain` example used in "NIS Domain" on page 25. Change the proxy agent ACI as it applies to your environment.

---

1. **Use ldapmodify to give proxy agent read permission for password by setting read ACI at base search DN if** `pam_unix` **is to be used on *all* clients for authentication.**

```
#ldapmodify -D "cn=Directory Manager" -w nssecret -f aci.ldif
```

   The contents of `aci.ldif` are:

```
dn: dc=mkt,dc=mainstore,dc=com
changetype: modify
add: aci
aci: (target="ldap:///dc=mkt,dc=mainstore,dc=com")
 (targetattr="userPassword")(version 3.0; acl "password read";
 allow (compare,read,search) userdn = "ldap:///cn=proxyagent,
 ou=profile,dc=mkt,dc=mainstore,dc=com"; )
```

2. **Use ldapsearch to see the new ACI setting.**

   `ldapsearch` shows the modified ACI:

```
#ldapsearch -L -h <servername> -b "dc=mkt,dc=mainstore,dc=com" \
  -s base "objectclass=*"
```

   The ACI returned by `ldapsearch` would look like:

```
dn: dc=mkt,dc=mainstore,dc=com
dc: mkt
associateddomain: mkt.mainstore.com
objectclass: top
objectclass: domain
objectclass: domainRelatedObject
objectclass: nisDomainObject
nisdomain: mkt.mainstore.com
aci: (target="ldap:///dc=mkt,dc=mainstore,dc=com")
 (targetattr="userPassword")(version 3.0; acl "password read";
 allow (compare,read,search) userdn = "ldap:///cn=proxyagent,
 ou=profile,dc=mkt,dc=mainstore,dc=com"; )
```

Because `pam_ldap` authentication is done at server side, there is no need to give read permission for password attribute to proxy agent. For information about `pam_ldap`, see "Pluggable Authentication Module (PAM)" on page 30.

▼ Convert NIS Data to LDIF Format.

If you are migrating from a NIS(YP) to an LDAP environment, use `dsimport` to convert NIS data into LDIF format. `dsimport` is part of NIS extension available on *iPlanet Advantage Software vol. 1* CD. You can access the documentation from the following web site:
`http://docs.iplanet.com/docs/manuals/directory.html`

1. Convert the NIS password data to LDIF format.

```
# cat passwd.nis | dsimport -n -m nis.mapping -t passwd \
 -M SIMPLE -D "" -w "" >passwd.ldif
```

Load the `passwd.ldif` file into the LDAP server.

2. Convert the NIS group data to LDIF format.

```
# cat group.nis | dsimport -n -m nis.mapping -t group \
-M SIMPLE -D "" -w "" > group.ldif
```

Load the `group.ldif` file into the LDAP server.

3. **Repeat the above step to convert all naming container files.**
4. **Use the ns-slapd `ldif2db` command or the `ldapadd` command to import the LDIF format files into the directory database.**

   For information about the `ns-slapd ldif2db` command, see "Managing Directory Server Databases" in the *Directory Server Administrator's Guide*. For information about `ldapadd`, see `ldapadd`(1)

Note - To convert file data to LDIF format, dsimport requires a modification to the mapping file to define how the entries are stored.

▼ Create Indexes to Improve Search Performance

Note - For information about how to create an index, see "Managing Indexes" in the iPlanet *Directory Server Administrator's Guide*.

---

1. **Index the following list of Solaris client attributes.**

```
membernisnetgroup    pres,eq,sub
nisnetgrouptriple    pres,eq,sub
memberuid            pres,eq
macAddress           pres,eq
uid                  pres,eq
uidNumber            pres,eq
gidNumber            pres,eq
ipHostNumber         pres,eq
ipNetworkNumber      pres,eq
ipProtocolNumber     pres,eq
oncRpcNumber         pres,eq
ipServiceProtocol    pres,eq
ipServicePort        pres,eq
nisDomain            pres,eq
nisMapName           pres,eq
mail                 pres,eq
```

2. **Use ldapsearch to determine if the directory supports Virtual List Views as identified by their OIDs; 1.2.840.113556.1.4.473 VLV control type and 2.16.840.1.113730.3.4.9 VLV control value.**

```
# ldapsearch -b "" -s base objectclass=\*
```

`ldapsearch` returns:

```
objectclass=top
namingcontexts=dc=sun,dc=com
namingcontexts=o=NetscapeRoot
subschemasubentry=cn=schema
supportedcontrol=2.16.840.1.113730.3.4.2
supportedcontrol=2.16.840.1.113730.3.4.3
supportedcontrol=2.16.840.1.113730.3.4.4
supportedcontrol=2.16.840.1.113730.3.4.5
supportedcontrol=1.2.840.113556.1.4.473
supportedcontrol=2.16.840.1.113730.3.4.9
supportedcontrol=2.16.840.1.113730.3.4.12
supportedsaslmechanisms=EXTERNAL
supportedldapversion=2
supportedldapversion=3
dataversion=atitrain2.east.sun.com:389 020000605172910
netscapemdsuffix=cn=ldap://:389,dc=atitrain2,dc=east,dc=sun,dc=com
```

3. **Index the following list of Virtual List View attributes.**

```
getpwent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getspent:      vlvFilter: (objectclass=posixAccount),     vlvScope: 1
getgrent:      vlvFilter: (objectclass=posixGroup),       vlvScope: 1
gethostent:    vlvFilter: (objectclass=ipHost),           vlvScope: 1
getnetent:     vlvFilter: (objectclass=ipNetwork),        vlvScope: 1
getprotoent:   vlvFilter: (objectclass=ipProtocol),       vlvScope: 1
getrpcent:     vlvFilter: (objectclass=oncRpc),           vlvScope: 1
getaliasent:   vlvFilter: (objectclass=rfc822MailGroup),  vlvScope: 1
getserviceent: vlvFilter: (objectclass=ipService),        vlvScope: 1
```

Create these indexes for any ou in the tree that contains a large number of objects or for those that are heavily accessed.

4. **For the password entry (getpwent), add the following entries to the directory.**

```
dn: cn=getpwent,cn=config,cn=ldbm
objectclass: top
objectclass: vlvSearch
cn: getpwent
vlvBase: ou=people,dc=eng,dc=sun,dc=com
vlvScope: 1
vlvFilter: (objectclass=posixAccount)
aci: (target="ldap:///cn=getpwent,cn=config,cn=ldbm")(targetattr="*")
 (version 3.0; acl "Config";allow(read,search,compare)userdn="ldap:///
anyone";)

dn: cn=getpwent,cn=getpwent,cn=config,cn=ldbm
cn: getpwent
vlvSort: cn uid
objectclass: top
objectclass: vlvIndex
```

5. Create the VLV index for getpwent.

```
# cd /usr/netscape/server4/slapd*
# ./vlvindex getpwent
OK# ./vlvindex getgrent
OK# ./vlvindex gethostent
OK# ./vlvindex getspent
OK#
# ./vlvindex
[05/Jun/2000:15:34:31 -0400] - ldbm2index: Unknown VLV Index named ''
[05/Jun/
2000:15:34:31 -0400] - ldbm2index: Known VLV Indexes are: 'getgrent',
'gethostent', 'getnetent', 'getpwent', 'getspent',
```

6. Repeat steps 4 an 5 for the rest of the Virtual List View attributes.

▼ Give "anyone" Read, Search, and Compare Permission on VLV Request Control

1. Use ldapsearch to show the VLV control ACI.

```
#ldapsearch -D "cn=Directory Manager" -w nssecret -b cn=features, \
cn=config objectclass=\*
```

The result of the search is:

```
cn=features,cn=config
objectclass=top
cn=features

cn=options,cn=features,cn=config
objectclass=top
cn=options

oid=2.16.840.1.113730.3.4.9,cn=features,cn=config
objectclass=top
objectclass=directoryServerFeature
oid=2.16.840.1.113730.3.4.9
cn=VLV Request Control
aci=(targetattr != "aci")(version 3.0; acl "VLV Request \
 Control"; allow( read,
search, compare ) userdn = "ldap:///all";)
```

2. **Use ldapmodify to give "anyone" read, search, compare permission for VLV feature. This ensures anonymous searches do not fail when trying to use VLV control.**

```
#ldapmodify -D "cn=Directory Manager" -w nssecret -f vlvcntrl.ldif
```

the contents of `vlvcntrl.ldif` are:

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

3. **Use ldapsearch to show the changes to the VLV control ACI.**

```
#ldapsearch -L -b "cn=features,cn=config" -s one \
oid=2.16.840.1.113730.3.4.9
```

The ACI returned by `ldapsearch` would look like:

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

▼ Add the proxyagent Entry to the LDAP Server

Note - This step is required only if a proxyagent entry is used.

1. **Add the proxyagent entry to the LDAP server.**

```
#ldapadd -D "cn=Directory Manager" -w nssecret -f proxyagent.ldif
```

The `proxyagent.ldif` file would look like:

```
dn: cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com
cn: proxyagent
sn: proxyagent
objectclass: top
objectclass: person
userpassword: proxy_agent_password
```

Note - The ou can be set to ou=profile or ou=person.

▼ Generate the Client Profile

1. **Generate the client profile and then add it to the LDAP server.**

```
ldap_gen_profile -P profile -b baseDN -D bindDN \
  -w bindDNpasswd ldapServer_IP_address(es)[:port#]
```

The `bindDN` is the bind DN of the proxy agent. You can specify more than one LDAP server's IP address if you want to allow fail over to another LDAP server. Capture the above result in a file, such as `profile.ldif`.

A typical command looks like:

```
ldap_gen_profile -P myProfile -b "dc=mkt,dc=mainstore,dc=com" \
-D "cn=proxyagent,ou=profile,dc=mkt,dc=mainstore,dc=com" \
-w proxy_agent_pswd -a simple 100.100.100.100 > profile.ldif
```

2. Add this client profile into LDAP server so that clients can download it.

CHAPTER 4

Client Setup

This chapter describes how to set up a Solaris client to use the LDAP name service.

Overview

When a Solaris client is made an LDAP client, it operates similar to a Solaris client using NIS / NIS+ or NFS. The client does hard lookups, which means the `getXXbyYY( )` calls wait until they get a response. Normally NIS(YP) has its servers on the local subnet (as they are normally bound to using a broadcast). Since Solaris 2.0 it has been possible (but not often used) to enable the use of NIS(YP) servers off the local subnet (see the `ypinit`(1M) command) and of course NIS+ is routinely setup without local servers. LDAP is more like NIS+ in it's tendency to deploy non-local servers.

This means that the routers become essential in making your machine work.

You must make sure your clients can always reach at least one of your LDAP servers. Either by making sure your network is properly reliable (most are unless someone cuts the wire or turns off the power to the router) or by making sure a server is on the local subnet (although again even then with subnet no longer being a cable but twisted pairs going to a ethernet server, cutting the cable or the power has the same effect).

The best method to keep your clients operational is to make sure you have multiple servers, keep those servers up to date (so they have the same data) and make sure your clients can reach all of them. Obviously if you are using the server preference feature (to force your clients to bind to certain servers) you need to make sure they meet the same criterion.

Fully Qualified Domain Name

One big difference between an LDAP client and a NIS or NIS+ client is that it always returns a FQDN (fully qualified domain name) (similar to those returned by DNS). For example, if your domain name is `engineering.example.net` and you lookup the hostname `server` with `getipnodebyname( )` (as they should in preparation for the conversion to IPv6 even though LDAP in this release only runs over IPv4). Both `gethostbyname( )` and `getipnodebyname( )` return the FQDN version `server.engineering.example.net`. Also if you use interface specific aliases like `server-#` you will see a long list of fully-qualified host names returned,

If you are using hostnames to share file systems or have other such checks you need to realize this key difference and account for it. Especially if you *assume* non-FQDN for local hosts and FQDN only for remote (DNS resolved) hosts. If you setup LDAP with a different domain name from DNS you might be surprised when the same host has two different FQDNs, depending on the lookup source.

ldap_cachemgr Daemon

The `ldap_cachemgr`(1M) is a daemon that runs on LDAP client machines. It refreshes the information in the configuration files from the LDAP server.

If `ldap_cachemgr` is not running, the configuration will not be updated.

Besides providing the refresh capability, the `ldap_cachemgr` provides a robust parsing mechanism that can flag any invalid syntax in the update query.

NIS / NIS+ to LDAP Transition

If you have upgraded a machine to SunOS 5.8 (Solaris 8) that was a NIS / NIS+ client and want to make it an LDAP client, run `ldapclient`(1M).

To run `ldapclient` you need to know the profile name and the IP address of at least one server. In the following example the profile name is `myprofile` and the LDAP server is at IP address `100.100.100.100` that runs on the default LDAP port number of `389`.

▼ Create an LDAP Client

1. **Become super user.**

2. **Run** `ldapclient`**(1M).**

```
# ldapclient -P myprofile 100.100.100.100
```

   `ldapclient` creates the configuration files and configures the client to use LDAP for name service lookups by modifying the `/etc/nsswitch.conf` file.

3. **Reboot the client.**

login to authenticate using ldap.

ldaplist Command

`ldaplist` is an LDAP utility to list the naming information from the LDAP servers. See `ldaplist`(1) for more info.

▼ List the Naming Information from the LDAP Servers

1. **List the containers for the baseDN.**

```
# ldaplist hosts myhost
dn: cn=myhost+ipHostNumber=100.100.100.100,ou=Hosts,
dc=mkt,dc=mainstore,dc=com
```

Without any argument, `ldaplist` returns all the containers in the current search `baseDN`.

APPENDIX **A**

Schemas

To support Solaris LDAP Naming clients, some Solaris specific schemas and some schemas defined by IETF are required.

This appendix has the following organization:

IETF Schemas

LDAP requires two schemas defined by IETF: the revised RFC 2307 Network Information Service schema and the LDAP mailgroups Internet draft.

RFC 2307 Network Information Service Schema

The LDAP servers must be configured to support the revised RFC 2307.

Note - Internet-Drafts are draft documents valid for a maximum of six months and might be updated, replaced, or obsoleted by other documents at any time

The nisSchema OID is 1.3.6.1.1. The RFC 2307 Attributes are:

```
( nisSchema.1.0 NAME 'uidNumber'
DESC 'An integer uniquely identifying a user in an
  administrative domain'
```

(continued)

(Continuation)

```
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.1 NAME 'gidNumber'
DESC 'An integer uniquely identifying a group in an
  administrative domain'
EQUALITY integerMatch SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.2 NAME 'gecos'
DESC 'The GECOS field; the common name'
EQUALITY caseIgnoreIA5Match
SUBSTRINGS caseIgnoreIA5SubstringsMatch
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.3 NAME 'homeDirectory'
DESC 'The absolute path to the home directory'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.4 NAME 'loginShell'
DESC 'The path to the login shell'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' SINGLE-VALUE )

( nisSchema.1.5 NAME 'shadowLastChange'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.6 NAME 'shadowMin'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.7 NAME 'shadowMax'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.8 NAME 'shadowWarning'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.9 NAME 'shadowInactive'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.10 NAME 'shadowExpire'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.11 NAME 'shadowFlag'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.12 NAME 'memberUid'
EQUALITY caseExactIA5Match
```

(continued)

(Continuation)

```
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String' )

( nisSchema.1.13 NAME 'memberNisNetgroup'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String' )

( nisSchema.1.14 NAME 'nisNetgroupTriple'
DESC 'Netgroup triple'
SYNTAX 'nisNetgroupTripleSyntax' )

( nisSchema.1.15 NAME 'ipServicePort'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.16 NAME 'ipServiceProtocol'
SUP name )

( nisSchema.1.17 NAME 'ipProtocolNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.18 NAME 'oncRpcNumber'
EQUALITY integerMatch
SYNTAX 'INTEGER' SINGLE-VALUE )

( nisSchema.1.19 NAME 'ipHostNumber'
DESC 'IP address as a dotted decimal, eg. 192.168.1.1
      omitting leading zeros'
SUP name )

( nisSchema.1.20 NAME 'ipNetworkNumber'
DESC 'IP network as a dotted decimal, eg. 192.168,
      omitting leading zeros'
SUP name SINGLE-VALUE )

( nisSchema.1.21 NAME 'ipNetmaskNumber'
DESC 'IP netmask as a dotted decimal, eg. 255.255.255.0,
       omitting leading zeros'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' SINGLE-VALUE )

( nisSchema.1.22 NAME 'macAddress'
DESC 'MAC address in maximal, colon separated hex
      notation, eg. 00:00:92:90:ee:e2'
EQUALITY caseIgnoreIA5Match
SYNTAX 'IA5String{128}' )

( nisSchema.1.23 NAME 'bootParameter'
DESC 'rpc.bootparamd parameter'
SYNTAX 'bootParameterSyntax' )
```

(continued)

(Continuation)

```
( nisSchema.1.24 NAME 'bootFile'
DESC 'Boot image name'
EQUALITY caseExactIA5Match
SYNTAX 'IA5String' )

( nisSchema.1.26 NAME 'nisMapName'
SUP name )

( nisSchema.1.27 NAME 'nisMapEntry'
EQUALITY caseExactIA5Match
SUBSTRINGS caseExactIA5SubstringsMatch
SYNTAX 'IA5String{1024}' SINGLE-VALUE )

( nisSchema.1.28 NAME 'nisPublicKey'
DESC 'NIS public key'
SYNTAX 'nisPublicKeySyntax' )

( nisSchema.1.29 NAME 'nisSecretKey'
DESC 'NIS secret key'
SYNTAX 'nisSecretKeySyntax' )

( nisSchema.1.30 NAME 'nisDomain'
DESC 'NIS domain'
SYNTAX 'IA5String' )
```

The nisSchema OID is 1.3.6.1.1. The RFC 2307 Objectclasses are:

```
( nisSchema.2.0 NAME 'posixAccount' SUP top AUXILIARY
  DESC 'Abstraction of an account with POSIX attributes'
  MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
  MAY ( userPassword $ loginShell $ gecos $ description ) )

( nisSchema.2.1 NAME 'shadowAccount' SUP top AUXILIARY
  DESC 'Additional attributes for shadow passwords'
  MUST uid
  MAY ( userPassword $ shadowLastChange $ shadowMin
        shadowMax $ shadowWarning $ shadowInactive $
        shadowExpire $ shadowFlag $ description ) )

( nisSchema.2.2 NAME 'posixGroup' SUP top STRUCTURAL
  DESC 'Abstraction of a group of accounts'
  MUST ( cn $ gidNumber )
  MAY ( userPassword $ memberUid $ description ) )

( nisSchema.2.3 NAME 'ipService' SUP top STRUCTURAL
  DESC 'Abstraction an Internet Protocol service.
        Maps an IP port and protocol (such as tcp or udp)
        to one or more names; the distinguished value of
        the cn attribute denotes the service's canonical
        name'
```

(continued)

(Continuation)

```
  MUST ( cn $ ipServicePort $ ipServiceProtocol )
  MAY ( description ) )

( nisSchema.2.4 NAME 'ipProtocol' SUP top STRUCTURAL
  DESC 'Abstraction of an IP protocol. Maps a protocol number
        to one or more names. The distinguished value of the cn
        attribute denotes the protocol's canonical name'
  MUST ( cn $ ipProtocolNumber )
  MAY  description )

( nisSchema.2.5 NAME 'oncRpc' SUP top STRUCTURAL
  DESC 'Abstraction of an Open Network Computing (ONC)
        [RFC1057] Remote Procedure Call (RPC) binding.
        This class maps an ONC RPC number to a name.
        The distinguished value of the cn attribute denotes
        the RPC service's canonical name'
  MUST ( cn $ oncRpcNumber $ description )
  MAY  description )

( nisSchema.2.6 NAME 'ipHost' SUP top AUXILIARY
  DESC 'Abstraction of a host, an IP device. The distinguished
        value of the cn attribute denotes the host's canonical
        name. Device SHOULD be used as a structural class'
  MUST ( cn $ ipHostNumber )
  MAY ( l $ description $ manager $ userPassword ) )

( nisSchema.2.7 NAME 'ipNetwork' SUP top STRUCTURAL
  DESC 'Abstraction of a network. The distinguished value of
        the cn attribute denotes the network's canonical name'
  MUST ipNetworkNumber
  MAY ( cn $ ipNetmaskNumber $ l $ description $ manager ) )

( nisSchema.2.8 NAME 'nisNetgroup' SUP top STRUCTURAL
  DESC 'Abstraction of a netgroup. May refer to other netgroups'
  MUST cn
  MAY ( nisNetgroupTriple $ memberNisNetgroup $ description ) )

( nisSchema.2.9 NAME 'nisMap' SUP top STRUCTURAL
  DESC 'A generic abstraction of a NIS map'
  MUST nisMapName
  MAY description )

( nisSchema.2.10 NAME 'nisObject' SUP top STRUCTURAL
  DESC 'An entry in a NIS map'
  MUST ( cn $ nisMapEntry $ nisMapName )
  MAY description )

( nisSchema.2.11 NAME 'ieee802Device' SUP top AUXILIARY
  DESC 'A device with a MAC address; device SHOULD be
        used as a structural class'
  MAY macAddress )

( nisSchema.2.12 NAME 'bootableDevice' SUP top AUXILIARY
```

(continued)

```
  DESC 'A device with boot parameters; device SHOULD be
  used as a structural class'
  MAY ( bootFile $ bootParameter ) )

( nisSchema.2.14 NAME 'nisKeyObject' SUP top AUXILIARY
  DESC 'An object with a public and secret key'
  MUST ( cn $ nisPublicKey $ nisSecretKey )
  MAY ( uidNumber $ description ) )

( nisSchema.2.15 NAME 'nisDomainObject' SUP top AUXILIARY
  DESC 'Associates a NIS domain with a naming context'
  MUST nisDomain )
```

Mail Alias Schema

The LDAP servers must be configured to support mail alias information. Mail alias information uses the schema defined by the LDAP Mailgroups Internet draft, formerly known as the `draft-steinback-ldap-mailgroups` draft. Until a new schema becomes available, Solaris LDAP clients will continue to use this schema for mail alias information.

Note - Internet-Drafts are draft documents valid for a maximum of six months and might be updated, replaced, or obsoleted by other documents at any time

---

The original LDAP Mailgroups schema contains a large number of attributes and object classes. Only two attributes and a single object class are used by Solaris clients. These are listed below

The mail alias Attributes are:

```
( 0.9.2342.19200300.100.1.3
  NAME 'mail'
  DESC 'RFC822 email address for this person'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String(256)'
  SINGLE-VALUE )

( 2.16.840.1.113730.3.1.30
  NAME 'mgrpRFC822MailMember'
  DESC 'RFC822 mail address of email only member of group'
  EQUALITY CaseIgnoreIA5Match
  SYNTAX 'IA5String(256)' )
```

The mail alias Objectclass is:

```
( 2.16.840.1.113730.3.2.4
  NAME 'mailGroup'
  SUP top
  STRUCTURAL
  MUST mail
  MAY ( cn $ mailAlternateAddress $ mailHost $ mailRequireAuth $
   mgrpAddHeader $ mgrpAllowedBroadcaster $ mgrpAllowedDomain $
   mgrpApprovePassword $ mgrpBroadcasterModeration $ mgrpDeliverTo $
   mgrpErrorsTo $ mgrpModerator $ mgrpMsgMaxSize $
   mgrpMsgRejectAction $ mgrpMsgRejectText $ mgrpNoMatchAddrs $
   mgrpRemoveHeader $ mgrpRFC822MailMember )
)
```

Solaris Schemas

The schemas required for the Solaris operating environment are the:

Extended user accounting schema.
Role based access control schema.
Solaris client naming profile schema.
projects schema

Extended User Accounting Schema

`/etc/user_attr` is a local source of extended attributes associated with users and roles. For more information see `user_attr`(4).

The extended user accounting Attributes are:

```
( 1.3.6.1.4.1.42.2.27.5.1.1 NAME 'SolarisProjectID'
  DESC 'Unique ID for a Solaris Project entry'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.2 NAME 'SolarisProjectName'
  DESC 'Name of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE )

( 1.3.6.1.4.1.42.2.27.5.1.3 NAME 'SolarisProjectAttr'
  DESC 'Attributes of a Solaris Project entry'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String )
```

(continued)

(Continuation)

```
( 1.3.6.1.4.1.42.2.27.5.1.30 NAME 'memberGid'
  DESC 'Posix Group Name'
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' )
```

The extended user accounting Objectclass is:

```
( 1.3.6.1.4.1.42.2.27.5.2.1 NAME 'SolarisProject'
  SUP top STRUCTURAL
  MUST ( SolarisProjectID $ SolarisProjectName )
  MAY ( memberUid $ memberGid $ description $ SolarisProjectAttr ) )
```

Role Based Access Control Schema

`/etc/user_attr` is a local source of extended attributes associated with users and roles. For more information see `user_attr`(4).

The role based access control Attributes are:

```
( 1.3.6.1.4.1.42.2.27.5.1.4 NAME 'SolarisAttrKeyValue'
  DESC 'Semi-colon separated key=value pairs of attributes'
  EQUALITY caseIgnoreIA5Match
  SUBSTRINGS caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.7 NAME 'SolarisAttrShortDesc'
  DESC 'Short description about an entry, used by GUIs'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.8 NAME 'SolarisAttrLongDesc'
  DESC 'Detail description about an entry'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.9 NAME 'SolarisKernelSecurityPolicy'
  DESC 'Solaris  kernel security policy'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.10 NAME 'SolarisProfileType'
  DESC 'Type of object defined in profile'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.11 NAME 'SolarisProfileId'
  DESC 'Identifier of object defined in profile'
```

(continued)

```
  EQUALITY caseExactIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.12 NAME 'SolarisUserQualifier'
  DESC 'Per-user login attributes'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.13 NAME 'SolarisReserved1'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )

( 1.3.6.1.4.1.42.2.27.5.1.14 NAME 'SolarisReserved2'
  DESC 'Reserved for future use'
  EQUALITY caseIgnoreIA5Match
  SYNTAX 'IA5String' SINGLE-VALUE )
```

The role based access control Objectclassses are:

```
( 1.3.6.1.4.1.42.2.27.5.2.3 NAME 'SolarisUserAttr' SUP top AUXILIARY
  DESC 'User attributes'
  MAY ( SolarisUserQualifier $ SolarisAttrReserved1 $ \
        SolarisAttrReserved2 $ SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.4 NAME 'SolarisAuthAttr' SUP top STRUCTURAL
  DESC 'Authorizations data'
  MUST cn
  MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisAttrShortDesc $ SolarisAttrLongDesc $ \
        SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.5 NAME 'SolarisProfAttr' SUP top STRUCTURAL
  DESC 'Profiles data'
  MUST cn
  MAY ( SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisAttrLongDesc $ SolarisAttrKeyValue ) )

( 1.3.6.1.4.1.42.2.27.5.2.6 NAME 'SolarisExecAttr' SUP top AUXILIARY
  DESC 'Profiles execution attributes'
  MAY ( SolarisKernelSecurityPolicy $ SolarisProfileType $ \
        SolarisAttrReserved1 $ SolarisAttrReserved2 $ \
        SolarisProfileId $ SolarisAttrKeyValue ) )
```

Solaris Client Naming Profile Schema

`/etc/user_attr` is a local source of extended attributes associated with users, roles, and profiles. For more information see `user_attr`(4).

`/etc/security/prof_attr` is a local source for execution profile names, descriptions, and other attributes of execution profiles. For more information see `prof_attr`(4).

The Solaris client naming profile Attributes are:

```
( 1.3.6.1.4.1.42.2.27.5.1.15 NAME 'SolarisLDAPServers'
  DESC 'LDAP Server address eg. 76.234.3.1:389'
  EQUALITY caseIgnoreIA5Match
  SYNTAX SolarisLDAPServerSyntax)

( 1.3.6.1.4.1.42.2.27.5.1.16
  NAME 'SolarisSearchBaseDN'
  DESC 'Search Base Distinguished Name'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.17
  NAME 'SolarisCacheTTL'
  DESC 'TTL value for the Domain information eg. 1w, 2d, 3h, 10m, or 5s'
  EQUALITY caseIgnoreMatch
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.18
  NAME 'SolarisBindDN'
  DESC 'DN to be used to bind to the directory as proxy'
  EQUALITY distinguishedNameMatch
  SYNTAX DN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.19
  NAME 'SolarisBindPassword'
  DESC 'Password for bindDN to authenticate to the directory'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.20
  NAME 'SolarisAuthMethod'
  DESC 'Authentication method to be used eg. "NS_LDAP_AUTH_NONE",
       "NS_LDAP_AUTH_SIMPLE" or "NS_LDAP_AUTH_SASL_CRAM_MD5"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.21
  NAME 'SolarisTransportSecurity'
  DESC 'Transport Level Security method to be used eg.
        "NS_LDAP_SEC_NONE" or "NS_LDAP_SEC_SASL_TLS"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.22
  NAME 'SolarisCertificatePath'
  DESC 'Path to certificate file/device'
  EQUALITY caseExactIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.23
```

(continued)

```
  NAME 'SolarisCertificatePassword'
  DESC 'Password or PIN that grants access to certificate.'
  EQUALITY caseExactIA5Match
  SYNTAX OctetString SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.24
  NAME 'SolarisDataSearchDN'
  DESC 'Search DN for data lookup in "<database>:(DN0),(DN1),..." format'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String)

( 1.3.6.1.4.1.42.2.27.5.1.25
  NAME 'SolarisSearchScope'
  DESC 'Scope to be used for search operations eg.
        "NS_LDAP_SCOPE_BASE", "NS_LDAP_SCOPE_ONELEVEL" or
       "NS_LDAP_SCOPE_SUBTREE"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.26
  NAME 'SolarisSearchTimeLimit'
  DESC 'Time Limit in seconds for search operations'
  EQUALITY integerMatch
  SYNTAX INTEGER SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.27
  NAME 'SolarisPreferredServer'
  DESC 'Preferred LDAP Server address or network number'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IAString)

( 1.3.6.1.4.1.42.2.27.5.1.28
  NAME 'SolarisPreferredServerOnly'
  DESC 'Boolean flag for use of preferredServer or not'
  EQUALITY booleanMatch
  SYNTAX BOOLEAN SINGLE-VALUE)

( 1.3.6.1.4.1.42.2.27.5.1.29
  NAME 'SolarisSearchReferral'
  DESC 'referral chasing option eg.
       "NS_LDAP_NOREF" or "NS_LDAP_FOLLOWREF"'
  EQUALITY caseIgnoreIA5Match
  SYNTAX IA5String SINGLE-VALUE)
```

The Solaris client naming profile Objectclasss is:

```
( 1.3.6.1.4.1.42.2.27.5.2.7 NAME 'SolarisNamingProfile'
  SUP top STRUCTURAL
  DESC 'Solaris LDAP Naming client profile objectClass'
  MUST ( cn $ SolarisLDAPServers $ SolarisSearchBaseDN )
  MAY ( SolarisBindDN $ SolarisBindPassword $ SolarisAuthMethod $
        SolarisTransportSecurity $ SolarisCertificatePath $
```

(Continuation)

```
        SolarisCertificatePassword $ SolarisDataSearchDN $
        SolarisSearchScope $ SolarisSearchTimeLimit $
        SolarisPreferredServer $ SolarisPreferredServerOnly $
        SolarisCacheTTL $ SolarisSearchReferral )
)
```

APPENDIX **B**

Troubleshooting the Configuration

This appendix describes configuration problems and suggested solutions.

Configuration Problems and Solutions

The following discussion briefly describes LDAP configuration problems and suggested solutions to the problems.

Unresolved Hostname

The Solaris LDAP client backend is designed to return fully qualified hostnames for host lookups, such as hostnames returned by `gethostbyname`(3N) and `getipnodebyname`(3N). If the name stored is fully qualified (that is contains at least one dot), the client returns the name as is. For example, if the name stored is `hostB.eng`, the returned name is `hostB.eng`.

If the name stored in the LDAP directory is not fully qualified (it does not contain any dot), the client backend appends the domain part to the name. For example, if the name stored is `hostA`, the returned name is `hostA.domainname`.

Unable to Reach Systems in the LDAP Domain Remotely

If the DNS domainname is different from the LDAP domainname, change the `nsswitch.conf` file. In the host entry, specify `dns`or put `dns` before `ldap`.

Sendmail Fails to Deliver/Receive Mail To/From Remote Users

If your mail domain (commonly the DNS domain) is different from the LDAP domain, you might run into a mail delivering problem. `sendmail`(1M) derives the mail domain from the domain portion of the hostname returned by `gethostname`(3N). This means the return address will be in the LDAP domain. Because the mail/DNS domain is different from the LDAP domain, external users cannot respond to the email To fix this problem, change the host entry in the `nsswitch.conf` file to `dns` or put `dns` before `ldap`.

Login Does Not Work

LDAP clients use the `PAM`(3) modules for user authentication during the logins. When using the standard unix `PAM` module, the password is read from the server and checked on the client side. This can fail due to one of the following reasons:

1. `ldap` does not exists as a source in the `/etc/nsswitch.conf` file
2. Password on the server is not readable by the proxy agent. You need to allow at least the proxy agent to read the password because the proxy agent returns it to the client for comparison
3. Incorrectly configured proxy agent causes authentication to fail.
4. The entry does not have the shadowAccount objectclass.

Lookup Too Slow

The LDAP database relies on indexes to improve the performance. A major performance degradation occurs when indexes are not configured properly. As part of the documentation, we have provided a common set of attributes that should be indexed. You can also add your own indexes to improve performance at your site.

`ldapclient` Cannot Bind to Server

`ldapclient` failed to initialize the client when using the `-P` profile option. There are several possible reasons for this failure

1. Check that the `ldap_cachemgr` is running (`ps -ef |grep ldap`) should show it running.
2. Try running `ldapclient -l` to check out the contents of the LDAP client cached files.

Note - Do not try to read the configuration and credential files directly as there is no guarantee they are in ASCII readable format.

---

3. `nisDomain` attribute is not set in the DIT to represent the entry point for the specified client domain.
4. Virtual list view indexing is not set up properly on the server.
5. Access control information is not set up properly on the server; thus disallowing anonymous search in the LDAP database.
6. Incorrect server address passed to the `ldapclient` command. Use `ldapsearch`(1) to verify the server address
7. Incorrect profile name passed to the `ldapclient` command. Use `ldapsearch`(1) to verify the profile name in the DIT.
8. Use `snoop`(1M) on the client's network interface to see what sort of traffic is going out, and determine to which server it is talking.

Index

D

E

F

G

I

L

M

N

R

S

U

V