

# NIS+ to LDAP Migration in the Solaris™ 9 Operating Environment

A Technical White Paper



# Table of Contents

<b>Executive Summary</b> .....	<b>.1</b>
<b>Introduction — NIS+ and LDAP</b> .....	<b>.3</b>
NIS+ .....	3
LDAP .....	4
<b>LDAP Overview</b> .....	<b>6</b>
Information Model .....	7
Naming Model .....	8
Directory Objects and Attributes .....	8
Directory Schema .....	8
Distinguished Names (DN) .....	9
Functional Model .....	9
Security Model .....	10
Replication .....	10
Native LDAP and the LDAP API .....	11
Native LDAP Commands .....	11
Command Line Tools for Using the LDAP API .....	11
Password Considerations .....	12
Pluggable Authentication Modules (PAM) .....	12
Solaris 9 OE LDAP Directory Services Overview .....	12
<b>Transition Tools</b> .....	<b>.13</b>
Overview .....	13
Transitioning from NIS+ to LDAP .....	13
NIS+ to LDAP Migration Example .....	14
<b>Solaris OE LDAP Directions</b> .....	<b>.17</b>
<b>Conclusion</b> .....	<b>.18</b>

## Chapter 1

# Executive Summary

NIS+ was designed as a replacement to NIS, providing naming services for large organizations. Its hierarchical architecture, strong user and workstation authentication, and incremental update capability is better suited than its predecessor for environments with tens of thousands of users.

More recently, LDAP directory services have emerged as an industry-standard way to consolidate and manage user and system information for very large organizations. LDAP directories are capable of becoming the central repository for all user, group, and access control information.

New in the Solaris™ 9 Operating Environment (OE) is an integrated LDAP directory service — the Sun™ ONE Directory Server 5.1 software. Previous versions of the Solaris OE included LDAP directory software, but it was copackaged (shipped in the box, but not otherwise integrated). Sun ONE Directory Server 5.1 software offers a number of advantages over NIS+ and other directory services, including:

- **Greater scalability:** The Sun ONE Directory Server can scale to millions of entries. Its architecture, performance, and administration tools support large-scale deployment.
- **Greater deployment flexibility:** Information can be stored and retrieved using industry-standard protocols. A centralized repository of user information can be used by virtually every application.
- **Superior availability:** The Sun ONE Directory Server offers multiple-master replication, providing fault tolerance and write-failover capability.
- **Standards-based security:** Enhanced authentication mechanisms offer greater security for enterprise applications and services. LDAP offers standards-based password mechanisms and encrypted transactions. Access control instructions (ACIs) enable fine-grain permission control

down to individual attributes in LDAP. As well, user changes and deletions can be automated to ensure that appropriate privileges are enforced across the organization.

- Lower cost: Compared with maintaining multiple naming services and directories across an organization, a centralized directory service helps cut costs by reducing duplication and overlapping efforts.

Any Solaris 9 OE system, including NIS+ clients, can use the LDAP API and tools built on the LDAP API, such as `ldapsearch(1)`, to access LDAP data. When a Solaris 9 OE system has been set up (using `ldapclient(1M)`) to use LDAP as a naming service — for user login, host name to address translation, and so on — it is referred to as a “Native LDAP” client. A system cannot be an NIS+ and Native LDAP client at the same time; in particular, a NIS+ server system cannot be made a Native LDAP client while it remains a NIS+ server.

Sun Professional Services personnel are available to assist organizations in their transition to an LDAP naming service, including all aspects of the transition such as design, test, and implementation.

This paper is intended to be a resource to system administrators who want to migrate from an NIS+ naming service environment to the LDAP directory service which is available in the Solaris 9 OE. The reader is assumed to be familiar with naming services. This document offers a brief overview of both the NIS+ naming service and LDAP directory services, as well as transitioning from NIS+ to LDAP. In addition, detailed information on the Sun ONE Directory Server 5.1 implementation in the Solaris 9 OE is included. Additional information on LDAP is available at [docs.sun.com](http://docs.sun.com).

## Chapter 2

# Introduction — NIS+ and LDAP

## NIS+

The NIS+ naming service enables administrators to store information about machine addresses, security information, mail information, Ethernet interfaces, and network services in central locations where all machines on a network are able to access it. This configuration of network information is referred to as the NIS+ namespace.

The NIS+ namespace is hierarchical, and is similar in structure to the UNIX® directory file system. The hierarchical structure allows a NIS+ namespace to be configured to conform to the logical hierarchy of an organization. The namespace's layout of information is unrelated to its physical arrangement. Thus, a NIS+ namespace can be divided into multiple domains that can be administered autonomously. Clients may access information in other domains in addition to their own, if they have the appropriate permissions.

NIS+ uses a client-server model to store and access the information contained in an NIS+ namespace. Each domain is supported by a set of servers. The primary server is called the master, and secondary servers are called replicas. Both master and replica servers run NIS+ server software, and both maintain copies of NIS+ tables. Changes made to the NIS+ data on the master server are propagated incrementally and automatically to the secondary servers.

To protect the structure of the namespace and its information, NIS+ includes a sophisticated security system. It uses authentication and authorization to verify whether a client's request for information should be fulfilled. Authentication determines whether the information requester is a valid user on the network. Authorization determines whether a particular user is allowed to have or modify the information requested.

## LDAP

Previous naming services, such as NIS+, were designed for specific purposes such as storing information about system resources and user accounts. Application access to NIS+ directories was achieved through APIs specifically created for this purpose. Applications used these APIs through procedure calls. This worked well in a configuration that relied solely on the Solaris OE, but required porting of relevant portions of the ONC+ software to platforms other than Sun™ platforms before the applications could use NIS+.

A more general-purpose directory service was needed, so CCITT and ISO (two standards bodies) developed a specification — the X.500 protocol. In addition to defining an access protocol, the X.500 specification also spelled out a set of rules for defining directory object names so they could be easily located. While the overall definition of X.500 was sound, the implementation was generally considered cumbersome to use.

CCITT and ISO then created another version of the X.500 specification, called the Lightweight Directory Access Protocol (LDAP), which enjoys wide industry acceptance. While LDAP naming services offer the same features and functionality as NIS+, they also offer compatibility across multiple applications in heterogeneous environments, flexible user access control, and incremental data pushes between master and replica servers. Overall, LDAP directories can provide the scalability and flexibility required in very large installations.

LDAP may be used to replace existing application-specific directories and consolidate information. LDAP-compliant systems can leverage a single master directory that controls all user, group, and access control information. A unified directory provides this information for all applications and systems on the network. This means that in a properly developed environment, directory-enabled applications can take advantage of changes made to an LDAP directory. For example, when adding information about a new user, administrators perform this task only once through a single interface. The user has immediate access to a UNIX user account, mail address and aliases, membership in departmental mailing lists, access to restricted Web servers, and inclusion in job-specific restricted newsgroups. In addition, the user could be instantly granted profiled access to other enterprise-specific services or applications.

**TABLE 1** NIS+ and LDAP Feature Comparison

	NIS+	LDAP
Hierarchical DIB1	X	X
Dynamic Updates2	X	X
Distributed DIB3	X	X
Extensible DIB4	X	X
Dynamic Replication5	X	X
Incremental Updates	X	X
Authentication	X	X
Access Control	X	X
Complex Data		X
Multiple-Master Replication		X

**TABLE 1** NIS+ and LDAP Feature Comparison

	<b>NIS+</b>	<b>LDAP</b>
Data storage	Multicolumn tables	Directories (varied)
Server Names	Master/Replica	Master/Replica
Security	Diffie-Hellman public key authentication	SSL, SASL/MD5
Implementation Scale	LAN	Global

## Table 1 Notes:

1. Hierarchical Directory Information Base (DIB) — The ability to organize the name space in a layered, tree-like structure.
1. Dynamic Updates — The ability to add, modify, and delete information in the name space and have those changes be immediately visible to users of the service.
2. Distributed Directory Information Base (DIB) — The ability to service the namespace from multiple nodes on the network.
3. Extensible Directory Information Base — The ability to dynamically expand the type of information stored as part of the namespace.
4. Dynamic Replication — The ability to dynamically propagate changes made to the DIB to other nodes that serve the DIB.

## Chapter 3

# LDAP Overview

LDAP is an interface standard for accessing a directory service. It is referred to as “lightweight” because is relatively easy to use and implement, especially when compared to X.500 protocol directory services, or services that use complex encoding mechanisms and require the use of special protocol stacks. LDAP runs on top of the TCP/IP protocol, enabling many operating systems to take advantage of its features and capabilities without requiring specialized client software, or proprietary stacks. Its industry-standard interfaces offer organizations a choice of which directory services they will use. A number of vendors provide LDAP directories, ensuring that users are not locked into a proprietary model.

In general, an LDAP directory is a repository of data that contains information about objects, such as information on people and resources. LDAP directories can also provide information relating to network services such as print services or other IT network resources. These entries are stored in a hierarchical (not relational) namespace capable of supporting large amounts of information.

LDAP-compliant systems can leverage a single master directory that contains all user, group, and access control information. LDAP directories may act as a central repository for information that is used by various systems, applications, and services for access control and fast data queries. Unifying directory information eliminates redundancy, which can help lower the total cost of ownership (TCO). In addition, a unified directory service can help applications run both inside and outside an organization. This enables participation by partners, customers, and suppliers in network applications.

The LDAP architecture may be more easily understood by examining the four models it supports:

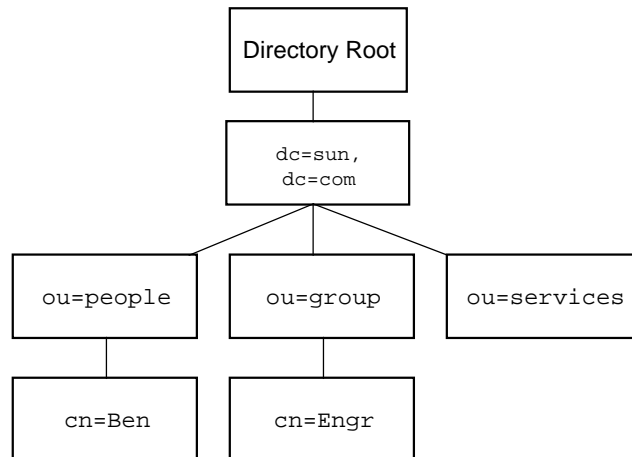
- Information: Defines how entries in the directory are organized
- Naming: Describes the syntax and structure of directory entries
- Functional: Describes the bind operation
- Security: Defines access rights for authenticated users

## Information Model

Entries are arranged in a tree-like structure called the directory information tree (DIT). A sample is shown in Figure 1. At the top of the DIT is the directory root, which is identified by the server name and port number on which the directory service is running. Multiple instances of the directory service can be running on the same server with each instance having its own DIT. Below the directory root is the directory suffix, of which there may be several per DIT. Suffixes can be expressed as an organization (`o=`) or as an Internet-style domain component (`dc=`). The domain-based format typically mirrors a company's DNS domain address and is expressed as domain component (`dc`) entries.

Located below the suffix are organization unit (`ou`) entries. These entries can be nested, so an `ou` may contain other organization units. The name chosen for an `ou` needs to be unique only at the level at which it resides, which means the same `ou` may be used in a different portion of the DIT without creating a conflict. An `ou` entry called `ou=people` is created during the default Sun ONE Directory Server installation. This entry is the default location for storing user account information, but any `ou` may be used for that purpose.

**Figure 4-1:** Sample directory information tree



If there are multiple directory servers in a network, they can be linked by LDAP referrals. A referral is a mechanism that instructs an LDAP client searching the directory to continue the search on another directory server. The referral accomplishes this instruction by passing a uniform resource locator (URL) back to the client. Once the client receives the URL, it can access the specified directory server.

## Naming Model

LDAP is designed to be flexible, but it also provides a structure so that LDAP clients can access data in any LDAP-compliant directory. For comparison purposes, an LDAP directory is unlike a Solaris OE file system, where a search can always be initiated from the root file system (/). Instead, an LDAP directory search begins by specifying an entry, such as `dc=blueprints,dc=com`, as a search base. The entry name is specified as a distinguished name (DN), which is part of a series of relative distinguished names (RDNs). Each directory server contains a single root directory specific entry (DSE), which contains basic information about the LDAP server. The DSE is specified during base-level searches on a directory when the name of a particular suffix is not known.

### Directory Objects and Attributes

The structure of a directory entry is defined by the object class to which it belongs. An object class defines a set of attributes that can be stored in a directory entry. LDAP object classes are extensible by creating a new class that is the child of an existing class. All the attributes defined in the parent class are inherited by the child. The name of an object class must be unique within the directory server and can be registered as a standard LDAP object. These objects are assigned a numeric object identifier (OID) to ensure they will not conflict with another object class.

Attribute names are unique within the directory server, and can be contained in more than one object class. The type of data that can be stored in an attribute is well defined, as is the way LDAP searches treat the data. For example, search matching for a string-valued attribute can be case-sensitive or insensitive. Attributes may also contain more than one value and can have aliases.

To promote interoperability, a set of standard LDAP object classes and attributes have been defined. Definitions of these ship with most LDAP servers in the form of schema configuration files. If they do not exist on a server, the content of these schema files can be added to the LDAP configuration files.

### Directory Schema

The information specified in a directory schema includes the object class name, required and allowed attributes, an optional OID number, and the allowable syntax. Table 2 shows the schema definition for the `posixAccount` object class attributes that stores the Solaris OE user account information.

**TABLE 2** Directory schema definition

Attribute	Description	Syntax
<code>cn(commonName)</code>	Common name of the POSIX account	<code>cn</code> (1-many)
<code>gidNumber</code>	Unique integer identifying group membership	<code>int</code> (single)
<code>homePhone</code>	The entry's home phone number	<code>tel</code>
<code>uid(userID)</code>	The user's login name	<code>cn</code> , (single)
<code>uidNumber</code>	Unique integer identifying a user	<code>int</code>

**TABLE 2** Directory schema definition

Attribute	Description	Syntax
<code>description</code>	A human-readable description of the object	<code>cis</code>
<code>gecos</code>	GECOS comment field	<code>cis</code>
<code>loginShell</code>	Path to the login shell	<code>ces (single)</code>
<code>userPassword</code>	Entry's password and encryption method	<code>bin, (single)</code>

In this example, `cn` is a case-insensitive string that can contain multiple values. The `gidNumber` and `uidNumber` are integers, and `homePhone` is represented by a special data type used for telephone numbers. Note that the LDAP `uid`, which is a string, is not the same as the numeric Solaris OE UID, which is represented by the LDAP attribute `uidNumber`.

### Distinguished Names (DN)

A directory entry is identified by its DN, which is similar to a file system path name. Entries are composed of many attributes, some of which are the same as other entries. To distinguish between entries that may have the same values for some attributes, one attribute is usually singled out as being unique. For user account entries defined in the `posixAccount` object class, that attribute is `uid`.

To prevent duplicate values being used, the Sun ONE Directory Server is configured by default to enforce `uid` attribute uniqueness. Entries that do not have a `uid` attribute are typically identified by the `commonName (cn)` attribute, which is available in most object classes, but is not required by all object classes such as `organization (o)` and `organization unit (ou)`.

The form of a DN is:

```
attribute=value,container,suffix
```

where there may be multiple containers depending on the DIT topology. An example of a DN for a user account is:

```
uid=cathy,ou=People,dc=blueprints,dc=com
```

The RDN specifies the left-most portion of the DN, which uniquely identifies the entry relative to its parent. For example:

```
uid=cathy
```

In this case, `uid=cathy` has to be unique within the `ou=People` container.

## Functional Model

Clients that need access to data on an LDAP server must first perform a bind operation. The bind operation requires, at a minimum, the DN of the user account entry with which the client wishes to bind. If the entry has a password, then (depending on the authentication method) it is passed along with the DN. Alternatively, the client can perform an anonymous bind, which does not require a particular user name or password.

The type of authentication the directory server requires is specified as part of the bind request. The default is simple authentication, which compares the password sent with the password stored for the specified DN. Since the simple authentication method sends a clear-text password, it is not secure unless it is combined with transport layer security (TLS), such as SSL. However, other authentication methods (for example, DIGEST-MD5) work securely even without the use of TLS.

If the bind operation is successful, the client is considered authenticated. All subsequent client requests made on the connection established as a result of the bind are performed as the authenticated user. After the LDAP client requests are complete, an unbind operation is performed to release the connection.

## Security Model

Access to LDAP entries on the server is protected by the rights established for the authenticated user. The rights can be assigned at the container, object, or attribute level. A portion of the DIT can be assigned stricter (or looser) control than other parts of the DIT. All entries of the same object class type can be assigned the same control. Control may also be established at the attribute level to protect certain information. For example, an employee's password might have restricted access, while other information is available to everyone.

The mechanism used to assign access rights is called the access control instruction (ACI). A single ACI can protect the entire DIT, or several may be used to provide finer-grained protection. When multiple ACIs are created, the ACI specifying deny access takes precedence. For example, if access is granted to everyone at the top level of the DIT but `ou=Contractors` are denied access, then the permissions set for `ou=Contractors` is enforced.

## Replication

Replication is the mechanism by which directory data is automatically copied from one directory server to another. Using replication, anything can be copied from entire directory trees to individual directory entries between servers.

Sun ONE Directory Server uses three types of replication schemes:

- **Single-Master Replication:** The master copy of directory data is held in a single master replica on the supplier server, which supplies updates to the consumer replica stored on a consumer server.
- **Master replica:** A read-write database that contains a master copy of the directory data. A master replica can process update requests from directory clients.
- **Multiple-Master Replication:** Sun ONE Directory Server 5.1 software also supports more complex replication scenarios in which the same information can be mastered on two servers. The two masters keep in sync with each other and provide fault tolerance. This information is held in a master replica on each server.
- **Cascading Replication:** A hub supplier server acts both as a consumer and supplier for a particular replica. It receives updates from the supplier server which holds the master copy of the data, and in turn supplies those updates to the consumer.

Note these differences in how replicas work in LDAP directory servers when compared with NIS+ naming services:

- In LDAP directory replication, two kinds of replicas are available:
  - (1) A read-write replica contains master copies of directory information and can be updated, while
  - (2) a read-only replica refers all update operations to read-write replicas.
- In NIS+, a replica is an NIS+ server that contain all or part of an NIS+ database. The data is read-only.

## Native LDAP and the LDAP API

It is important to understand the difference between Native LDAP and the LDAP API. A Solaris OE system that has been configured to use LDAP as a naming service — for user login, machine name to address translation, and other actions — is a Native LDAP client. The Solaris OE also provides an LDAP API (see `ldap(3LDAP)`), and a number of tools, including `ldapsearch(1)`, `ldapadd(1)`, `ldapdelete(1)`, `ldapmodify(1)`, and `ldapmodrdn(1)` that are built directly on top of the API. These tools, and other applications that use the LDAP API, are independent of Native LDAP. The tools can be used on systems that are clients of other naming services, such as NIS+. Note that a system cannot be both a NIS+ client and a Native LDAP client at the same time.

### Native LDAP Commands

Two commands are useful when preparing an LDAP environment:

- `ldapclient` initializes a system to be a Native LDAP client and use LDAP as a repository for naming service data. The `ldapclient` utility may also be used to restore the network service environment on Native LDAP clients, and to list the contents of the LDAP client cache in human-readable format.
- `idsconfig` prepares the Sun ONE Directory Server to be populated with data and serve Native LDAP clients. Administrators may specify the input configuration file with the `-i` option on the command line. Alternatively, the tool can prompt the user for configuration information. The input configuration file is created by `idsconfig` with the `-o` option on a previous run.

### Command Line Tools for Using the LDAP API

The following command line tools may also be useful in a NIS+-to-LDAP migration. The tools correspond to the operations performed by the LDAP API. Each 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

More information on the LDAP command line tools can be found on the respective man pages. For more on the LDAP API, see `ldap(3LDAP)` in the man pages.

## Password Considerations

Note that while transitioning from NIS+ to LDAP, only NIS+ password capabilities can be used. These include:

- The elapsed time since the password was last modified
- Minimum number of days between password changes
- Maximum number of days the password is valid
- The number of days before password expiration that the user is warned
- Number of inactive days allowed for a user
- Absolute date when password expires

## Pluggable Authentication Modules (PAM)

Pluggable authentication modules (PAM) provide a way for applications to remain independent of the authentication scheme used in the Solaris 9 OE. 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.

With the initial release of Solaris 9 OE, as of this printing, these password features are not available with a Native LDAP client. They are expected to be included in a future update.

To use PAM in an LDAP naming service, one of two PAM modules can be configured in `pam.conf`: `pam_unix_*` and `pam_ldap`. `pam_unix_*` provides the traditional model of UNIX authentication. `pam_ldap` authenticates users directly to the directory, enabling Solaris OE clients to work with newer, more advanced authentication methods that might be supported by the directory server. Note that accounts shared between NIS+ and LDAP must store the password in UNIX crypt format.

Additional information on using `pam_ldap` in the Solaris 9 OE can be found at [www.sun.com](http://www.sun.com).

## Solaris 9 OE LDAP Directory Services Overview

The Solaris 9 OE provides an integrated version of the powerful, distributed Sun ONE Directory Server software, which is designed to manage an enterprise-wide directory of users and resources. This scalable LDAP directory service can be used for intranet applications, extranets with trading partners, and e-commerce applications to reach customers over the Internet.

The Sun ONE Directory Server is managed through the Sun ONE Console GUI. Administrators use the console to grant access rights, manage databases, configure the directory, and replicate the data to multiple directory servers. Users access the data through any LDAP-enabled client application, such as those developed with the Sun ONE LDAP SDKs for C and the Java™ programming language.

There is support for the Solaris Package Format. Sun ONE Directory Server can be installed as part of a standard installation, using tools such as Solaris JumpStart™ software. Both 32-bit and 64-bit environments are supported.

## Chapter 4

# Transition Tools

Transition tools assist system administrators in transitioning from a NIS+ naming service environment to the more comprehensive directory services now available in the Solaris 9 OE. These tools provide gateway services, enabling an existing NIS+ service to use a back end LDAP directory, as well as transition tools, which enable existing NIS+ data to be ported into an LDAP directory service.

## Overview

A number of approaches may be used when transitioning from NIS+ to LDAP using the Solaris 9 OE. The Solaris 9 OE offers a number of capabilities that assist in transitioning from NIS+ to LDAP. Using the procedure outlined below, the existing NIS+ data can be automatically uploaded and maintained in an LDAP data store. In addition, only the NIS+ master server is modified — all other NIS+ clients or replicas remain unchanged.

This solution enables a smoother transition from NIS+ to LDAP. Administrators can first migrate naming service data to LDAP, then move clients to LDAP naming services as time permits.

## Transitioning from NIS+ to LDAP

An overview of the transition process:

- The Solaris 9 `rpc.nisd(1M)` can be configured to use an LDAP server as its data repository. In this way, the normal NIS+ database becomes a cache for the LDAP data, with configurable time to live (TTL) values. The cache improves performance, and enables the `rpc.nisd` to serve NIS+ data even if communication with the LDAP server is interrupted temporarily.

- Two configuration files, `rpc.nisd(4)` and `NIS+LDAPmapping(4)`, control communication between `rpc.nisd` and the LDAP server, as well as the way that NIS+ table entries are mapped to LDAP entries. A template configuration file, `/var/nis/NIS+LDAPmapping.template`, covers all standard NIS+ tables, and can be used as a basis when creating a customized NIS+ to LDAP mapping.
- A test utility, `nislldapmptest(1M)`, can be used to try out mapping configurations without affecting NIS+ data.
- Since the NIS+ `passwd.org_dir` table stores passwords in the UNIX crypt format, the LDAP server must be set up to use crypt format for the `userPassword` attribute for those accounts that are shared with NIS+.
- In general, migration from NIS+ to LDAP starts by installing and configuring one or more LDAP servers to support the Solaris OE name service clients.
- The next steps would be to install the Solaris 9 OE on the NIS+ master, and configure `rpc.nisd` on the NIS+ master to map NIS+ data to and from LDAP. The `rpc.nisd` offers options that enable uploading all NIS+ data to LDAP.
- Only the NIS+ master needs to run the Solaris 9 OE. All other NIS+ servers and clients can remain on earlier Solaris software releases (or whatever OS they are currently using).
- Once a complete LDAP name service environment exists, conversion of NIS+ clients to LDAP clients can start. While this conversion is going on, NIS+ and LDAP name service clients share the same data, and the NIS+ master `rpc.nisd` keeps the data in sync.

NIS+ servers remain NIS+ clients until their NIS+ server role is decommissioned, at which time they may be converted to LDAP name service clients. Once the NIS+ master is the only remaining NIS+ client, it can also be converted to be an LDAP client, completing the conversion.

## NIS+ to LDAP Migration Example

To make this a more complete example, without becoming bogged down in alternate migration routes, the following assumptions are made:

- An LDAP name service environment has been created. See the relevant Solaris OE documentation for how to use Sun ONE Directory Server 5.1 as included in the Solaris 9 OE, and especially the `idsconfig(1M)` command for how to prepare the Sun ONE Directory Server to serve LDAP name service clients.
- The existing NIS+ environment consists of a single NIS+ domain, with one NIS+ master, zero or more NIS+ replica servers, and a number of NIS+ clients systems.
- The LDAP name service environment is initially empty (no name service data), and will be initialized with NIS+ data.
- The NIS+ environment is maintained, and as much as possible is unchanged. This means that only the NIS+ master needs to be updated to the Solaris 9 OE, and serves as the sole LDAP gateway.

Before proceeding, make sure to back up all NIS+ data; see `nisbackup(1M)`.

**1.** Install the Solaris 9 OE on the NIS+ master.

**5.** Select an LDAP server that will serve as the repository for the NIS+ data.

- This LDAP server should be accessible over a low-latency/high-bandwidth connection from the NIS+ master.

- The server should be deployed using high-availability technology, so it can reliably serve directory information to all applications and services.

Note that the LDAP server can be the same system as the NIS+ master.

6. Identify the authentication method to be used between the `rpc.nisd` and the LDAP server. Depending on the authentication method, identify an appropriate LDAP bind DN and password/key. The LDAP bind DN must have sufficient capability to enumerate the largest LDAP name service data container.
7. Edit `/etc/default/rpc.nisd` to reflect the chosen LDAP server address, authentication method, proxy user (bind DN), and password. See `rpc.nisd(4)` for syntax. Note that editing `/etc/default/rpc.nisd` does not, on its own, start mapping of data between NIS+ and LDAP.
8. Make a copy of the `/var/nis/NIS+LDAPmapping.template` file. This copy may have any name except `/var/nis/NIS+LDAPmapping`, since its presence would start NIS+ to LDAP mapping when the `rpc.nisd` is restarted — which would be premature at this point. Assume a copy of the mapping file is `/var/nis/nlm`.
9. Read the “Getting Started” section on `NIS+LDAPmapping(4)`, and customize the configuration files accordingly. In this example, only the NIS+ master maps LDAP data, so mapping of directory, table, and group objects should be disabled by commenting out (or removing) the relevant `nisplusLDAPdatabaseIdMapping`, `nisplusLDAPentryTtl`, and `nisplusLDAPobjectDN` attributes from `/var/nis/nlm`.

The relevant database IDs are `basedir`, `orgdir`, `groupsdir`, and `admin`, as well as all those database IDs that end with `_table`. This also means that the `nisplusObjectContainer` object class and `ou=nisPlus` container need not be created.

10. Edit the `defaultSearchBase` in `/etc/default/rpc.nisd` to reflect the base DN of the LDAP name service containers.
11. Test the mapping that has been created using the `nisldapmaptest(1M)` utility. Since a nonstandard name is being used for the mapping file, specify `-m nlm`. Remember to remove any non-NIS+ data that has been created with `nisldapmaptest` before proceeding to the next step.

12. Upload all NIS+ data to LDAP. To do this, terminate `rpc.nisd`, and then execute:

```
/usr/sbin/rpc.nisd -D \  
-x nisplusLDAPinitialUpdateAction=to_ldap \  
-x nisplusLDAPinitialUpdateOnly=yes
```

The NIS+ data is unaffected by this operation. While checking the result of the upload (using `nisldapmaptest`, `ldapsearch`, and so on), restart `rpc.nisd` to provide normal NIS+ service.

This is a good time to double check that backups exist of the NIS+ data; see `nis-backup(1M)`.

13. Rename `/var/nis/nlm` to `/var/nis/NIS+LDAPmapping`. Restart `rpc.nisd`, which now will use the LDAP server as its data repository.
14. Begin converting NIS+ client machines to become LDAP name service clients; see the `ldap-client(1M)` command. Of course, new systems may also be added that were LDAP name service clients from the start.
15. As the number of NIS+ clients decreases, it may be possible to decommission NIS+ replica servers. To do this, use the `nisrmdir -s <replica>` command to remove the replica from the

server list for NIS+ directories. When a NIS+ replica no longer serves any NIS+ directories, it can be converted to be an LDAP name service client.

**16.** Finally, when the only remaining NIS+ client is the NIS+ master itself, the NIS+ master can be converted to become an LDAP name service client.

**17.** Somewhere after step 10, but before step 13, perform these steps:

- Convert any applications that use the NIS+ API to be independent of NIS+. For example, convert those applications to use the LDAP API (see `ldap(3LDAP)`).
- Change admin procedures so that data from new user accounts, IP addresses, and so on are added to LDAP instead of NIS+.

For more information on setting up NIS+ to LDAP mapping, see “Transitioning from NIS+ to LDAP” in *Naming and Directory Services (FNS and NIS+)*, and the `NIS+LDAPmapping(4)` man page.

## Chapter 5

# Solaris OE LDAP Directions

Future versions of the Solaris OE are expected to contain additional directory service enhancements to the features and capabilities provided in the Solaris 9 OE. Tighter integration between subsequent releases of the Solaris 9 OE and the Sun ONE Directory Server is expected to provide a number of enhancements in these areas:

- **Security:** Stronger security can be enabled through use of the GSS-API, Kerberos, new authentication and encryption techniques, and Solaris PKI integration.
- **Administration:** New GUI tools can provide a more comprehensive administration platform.
- **Availability and scalability:** Enhanced replication capabilities can offer more services to more users.
- **Improved performance:** This could lower response times while raising the number of users or capabilities that each instance can service.
- **Additional API access:** Lower-level access can provide new application functionality.

## Chapter 6

# Conclusion

Sun is committed to the LDAP protocol as its naming service for the future. The Sun ONE Directory Server offers Native LDAP functionality for the Solaris 9 OE, and features superior scalability, flexibility in deployment, and strong support for industry standards. Migrating from the NIS+ naming service environment to Native LDAP in the Solaris OE offers many benefits. LDAP directories can:

- Serve as a common repository for user identity and management. A streamlined identity infrastructure can reduce overall administrative work.
- Provide a building block for addressing future requirements, such as service-on-demand initiatives.
- Deliver the ability to reuse an existing identity repository for new applications, instead of creating a new one. This can contribute to an increased return on investment (ROU) and faster time to service.

By integrating Sun ONE Directory Server into the Solaris 9 OE, Sun is able to deliver a naming and directory infrastructure that provides a centralized repository for user management and system configuration data. This information can be accessed through industry-standard protocols, making it available throughout the organization. The result is a competitive total cost of ownership with a network-wide application quality of service.

Copyright 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, California 94303 U.S.A. All rights reserved.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the U.S. and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, docs.sun.com, and Solaris JumpStart are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

RESTRICTED RIGHTS: Use, duplication, or disclosure by the U.S. Government is subject to restrictions of FAR 52.227-14(g)(2)(6/87) and FAR 52.227-19(6/87), or DFAR 252.227-7015(b)(6/95) and DFAR 227.7202-3(a).

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 901 San Antonio Road, Palo Alto, Californie 94303 Etats-Unis. Tous droits réservés.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, Java, Solaris, docs.sun.com, et Solaris JumpStart sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays. Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc.

CETTE PUBLICATION EST FOURNIE "EN L'ETAT" ET AUCUNE GARANTIE, EXPRESSE OU IMPLICITE, N'EST ACCORDEE, Y COMPRIS DES GARANTIES CONCERNANT LA VALEUR MARCHANDE, L'APTITUDE DE LA PUBLICATION A REpondre A UNE UTILISATION PARTICULIERE, OU LE FAIT QU'ELLE NE SOIT PAS CONTREFAISANTE DE PRODUIT DE TIERS. CE DENI DE GARANTIE NE S'APPLIQUERAIT PAS, DANS LA MESURE OU IL SERAIT TENU JURIDIQUEMENT NUL ET NON AVENU.



Please  
Recycle



Adobe PostScript

Sun Microsystems, Inc. 901 San Antonio Road, Palo Alto, CA 94303-4900 USA Phone 800 786-7638 or +1 512 434-1577 Web [sun.com](http://sun.com)



We make the net work.

**Sun Worldwide Sales Offices:** Africa (North, West and Central) +33-13-067-4680, Argentina +5411-4317-5600, Australia +61-2-9844-5000, Austria +43-1-60563-0, Belgium +32-2-704-8000, Brazil +55-11-5187-2100, Canada +905-477-6745, Chile +56-2-3724500, Colombia +571-629-2323, Commonwealth of Independent States +7-502-935-8411, Czech Republic +420-2-3300-9311, Denmark +45 4556 5000, Egypt +202-570-9442, Estonia +372-6-308-900, Finland +358-9-525-561, France +33-134-03-00-00, Germany +49-89-46008-0, Greece +30-1-618-8111, Hungary +36-1-489-8900, Iceland +354-563-3010, India-Bangalore +91-80-2298989/2295454; New Delhi +91-11-6106000; Mumbai +91-22-697-8111, Ireland +353-1-8055-666, Israel +972-9-9710500, Italy +39-02-641511, Japan +81-3-5717-5000, Kazakhstan +7-3272-466774, Korea +82-2-2193-5114, Latvia +371-750-3700, Lithuania +370-729-8468, Luxembourg +352-49 11 33 1, Malaysia +603-21161888, Mexico +52-5-258-6100, The Netherlands +00-31-33-45-15-000, New Zealand-Auckland +64-9-976-6800; Wellington +64-4-462-0780, Norway +47 23 36 96 00, People's Republic of China-Beijing +86-10-6803-5588; Chengdu +86-28-619-9333; Guangzhou +86-20-8755-5900; Shanghai +86-21-6466-1228; Hong Kong +852-2202-6688, Poland +48-22-8747800, Portugal +351-21-4134000, Russia +7-502-935-8411, Singapore +65-6438-1888, Slovak Republic +421-2-4342-94-85, South Africa +27 11 256-6300, Spain +34-91-596-9900, Sweden +46-8-631-10-00, Switzerland-German 41-1-908-90-00; French 41-22-999-0444, Taiwan +886-2-8732-9933, Thailand +662-344-6888, Turkey +90-212-335-22-00, United Arab Emirates +9714-3366333, United Kingdom +44-1-276-20444, United States +1-800-555-9SUN or +1-650-960-1300, Venezuela +58-2-905-3800