



Understanding Solaris™ 9 Operating Environment Directory Services

Tom Bialaski, Intregrated Products Group

Sun BluePrints™ OnLine—December 2002



<http://www.sun.com/blueprints>

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95045 U.S.A.
650 960-1300

Part No. 817-0776-10
Revision 1.0, 12/4/02
Edition: December 2002

Copyright 2002 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95045 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

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 United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, Sun ONE, iPlanet, Java, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the US and other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the Far and its supplements.

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., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

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 enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, Sun ONE, iPlanet, Java, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, 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.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please
Recycle



Adobe PostScript

Understanding Solaris 9™ Operating Environment Directory Services

The Network Information System (NIS) and its enhanced version, NIS+, have been around for over ten years. NIS is in wide use today and has served its purpose well. Due to the many advances that have taken place in standards-based directory technology in recent years, Sun Microsystems, Inc. has tried to leverage these advances with the introduction of the next generation of Solaris™ Operating Environment (Solaris OE) naming services.

The Solaris 8 OE provided a back-end to the Name Service Switch (NSS) that allowed Solaris OE clients to obtain naming information from a Lightweight Directory Access Protocol (LDAP) enabled directory service. This implementation is commonly referred to as the Solaris OE LDAP Client. The Solaris 9 OE improves this implementation in several areas, including the following:

- Ease of setup and configuration
- Flexible directory configuration options
- Enhanced security model

Because improved security features are the most important area of improvement, this implementation is commonly referred to as the Secured LDAP Client. This article examines the differences between the LDAP Client and the Secured LDAP Client, and explains how they can be supported on the same directory server. In addition, the article details troubleshooting tips for common implementation problems.

This article contains the following sections:

- “NIS and NIS+ Overview” on page 2
- “LDAP as a Naming Service” on page 3
- “Solaris 8 OE LDAP Implementation” on page 5
- “Solaris 9 OE Secured LDAP Client and Server Configuration” on page 9

NIS and NIS+ Overview

The premise behind any naming service is simple, provide a mechanism to translate one representation of data into another representation of data. The benefits of doing this are twofold. Easy to remember names can be assigned to numeric data, and changes can be made to one format without changing the other. For example, a script or application might reference the host name of a computer rather than its Internet Protocol (IP) address, allowing the computer to be moved to a new subnet and be assigned a new IP address without requiring you to modify the script or application.

Although the objective of a naming service might be simple, you must consider several complex issues. You should address the following questions and consider how NIS and NIS+ addresses them.

- Where and how is data centrally stored?

NIS addresses this issue by storing data in a binary format (that could be created from text files), then placing it on a server. The structure of the data is a simple *key-value* pair. To retrieve data, clients specify a key, such as a host name, then receive the associated value, such as an IP address. NIS+ improves data storage by supporting tables that can be accessed as rows and columns.

- How can clients easily access data?

NIS and NIS+ clients access naming service data through the transport independent remote procedure call (TI-RPC), which is commonly called the Sun RPC. These calls are defined in the Open Network Computing (ONC) specification for NIS and ONC+ for NIS+, implying that client platforms that want to access NIS or NIS+ have libraries that support these specifications ported to them.

- How can you eliminate single points of failure?

Single points of failure are eliminated by supporting multiple NIS/NIS+ servers that can supply the same data to clients. Data is synchronized by replication between servers. NIS replicates complete maps, or databases, while NIS+ replicates just the updates. Only one NIS or NIS+ server can function as a master at any time. That is, only one server can accept updates.

- How can you prevent unauthorized access?

For simplicity, NIS was designed to allow anonymous access from clients. Because of this, NIS servers typically accept connections from any client, as long as it belongs to the NIS domain of the server. You can modify this behavior by specifying a list of IP addresses of clients that are permitted access, and then restricting access to only those clients. However, with IP spoofing, this

mechanism can easily be circumvented. NIS+ implements additional security by providing the option to assign network passwords to NIS+ clients and requiring the password to be sent along with requests.

- Can you protect sensitive data from snooping?

NIS and NIS+ both send data, except for passwords, over the network in clear text. This makes the data susceptible to snooping. At the time NIS and NIS+ were developed, most naming service data was considered public and the overhead of encryption made deploying it restrictive.

- Does the naming service scale to an enterprise-wide level?

The NIS and NIS+ naming services scale horizontally by adding more servers as needed. Although NIS clients can access NIS servers that reside on remote subnets, a common configuration is to have at least one server per subnet, which reduces network traffic through routers.

- Can it interoperate with other naming services?

Because NIS is based on the ONC specification, it interoperates with other vendor implementations that adhere to the specification. That is, you can mix and match master and slave servers from different vendors and support NIS clients that run on different operating environments.

The Solaris 9 OE provides synchronization between NIS+ and LDAP with the introduction of an enhanced `rpc.nisd` process.

LDAP as a Naming Service

Designing a naming service based on existing industry standards presents its own set of challenges. Unlike NIS and NIS+, which were created for a specific purpose, directory servers are meant to be general purpose. While the flexibility they provide allows for a great deal of customization, there are a number of constraints imposed that are not present in proprietary (closed) implementations. The following paragraphs explain how the Solaris OE LDAP Client addresses the same issues presented in the previous section.

- Where and how is data centrally stored?

LDAP data is stored in a complex structure defined by the directory server schema. To promote interoperability between LDAP implementations, schema definitions are often proposed in Request for Comments (RFCs) and adopted as

standards. Object classes and attributes defined in the standard schema are assigned a unique identifier so they can be accessed by the same name no matter what implementation is deployed.

To map NIS and NIS+ data to an LDAP structure, the object classes and attributes defined in the RFC2307bis draft specification are used. For example, the `posixAccount` object class defines a structure to hold the fields present in `passwd` database entries. Likewise, the `ipHost` object class defines a structure to hold data from the `hosts`, `ipnodes`, and `bootparams` databases.

- How can clients easily access the data?

Clients access the directory server through LDAP, which runs on top of Transmission Control Protocol/Internet Protocol (TCP/IP). Unlike NIS or NIS+, no remote procedure call (RPC) programs are required. The libraries that support LDAP functions are available on most any platform.

- How can you eliminate single points of failure?

Directory servers have built-in replication facilities. Depending on the implementation, replication can be single- or multi-master. Single-master replication functions are similar to NIS and NIS+, allowing updates to be performed only on one server. Multi-master replication allows updates to be processed by more than one server. Only updates are propagated from a master to a replica, or a slave server. A change log on the master server maintains a record of all updates and is replayed through LDAP on the replicas to keep them synchronized.

Directory servers can also run in clusters to increase availability.

- How can you prevent unauthorized access?

LDAP directory servers provide flexible mechanisms for both authentication and authorization. For clients to authenticate to a server, they must support the same authentication method. In other words, just because a server supports an authentication method, such as DIGEST-MD5, a client cannot automatically use it. The Sun™ ONE Directory Server 5.1 (formerly known as iPlanet™ Directory Server) supports the Simple Authentication and Security Layer (SASL) framework, which allows the server to negotiate authentication mechanisms if the other party also supports SASL.

Authorization, or the determination of what entries the user has access rights to, is set on the directory server. Anonymous access can be granted, or the user might be required to furnish some form of credentials before being authenticated. Once authenticated, access control lists (ACLs) are used to determine what level of permission to grant the user for particular entries, or groups of entries.

Proxy authentication allows operations to be performed on behalf of a user by a proxy user. This is a useful technique for allowing operations to be performed that the user normally would not have permission to perform.

- How can you protect sensitive data from snooping?

Most directory servers provide a Secure Socket Layer (SSL) or Transport Layer Security (TLS) implementation that can be used for authentication and for data encryption. This is the same technology that is used to secure web transactions over the Internet. SSL/TLS can be configured so that the client trusts the server or the server trusts the client. As is the case with any authentication method, both client and server must support the same revision level of SSL/TLS to operate.

LDAP v3 compliant directory servers offer SSL/TLS through a protected port (636) or through the use of the Start TLS Extension (RFC 2830).

- Does the naming service scale to an enterprise-wide level?

Directory servers, by nature, have the ability to handle very large databases. It is not uncommon to maintain a million or more entries in a single directory server. These servers are also optimized for fast read access and can support thousands of read requests per second on typical midrange servers. Data distribution over multiple servers is another way directory services can scale.

Directory server scaling is generally achieved by deploying replicated servers with load-balancing switches to direct LDAP requests from clients to them, and also increases the availability of the service.

- Can it interoperate with other naming services?

Directory services can interoperate with other naming services that support LDAP v3. However, the way clients are required to authenticate to a naming service and how they gain authorization to network resources can be a limiting factor.

Solaris 8 OE LDAP Implementation

Technically speaking, the Solaris 8 LDAP implementation is client side only. In theory, because it communicates over the standard LDAP v3, any v3-compliant directory server should be able to support Solaris 8 OE LDAP clients. In practice, there are several required server features and configuration parameters that need to be set up. Because the set up is rather complex, scripts are available for the Netscape Directory Server 4.1x and Sun ONE Directory Server 5.1 software (formerly known as the iPlanet™ Directory Server). These are the only two directory servers that Sun Microsystems, Inc. supports.

Several design decisions were made in the development of the Solaris 8 OE LDAP Client that affect its implementation. The following sections detail some of these designs to illustrate why so many configuration changes on the directory server are necessary.

LDAP Client Profiles

There are a number of configuration parameters that a Solaris OE LDAP client needs to set before it can communicate with a directory server. Specific parameters include the address of the directory server(s) the client will attempt to get naming service data from and the name of a proxy account to bind to the server.

One approach you can use is to set these parameters in a local file on each client and update them locally. The obvious disadvantage to this approach is that more complex installations and cumbersome updates are required if the configuration parameters need to be changed. Alternatively, you can maintain the configuration data in a central repository, like the directory itself, so it can be easily retrieved during client installation and reconfiguration. This data is stored in client *profiles*.

Storing client configurations in the directory requires you to create schema definitions that define the object class and attributes that contain the client profile data. This requires the extension of the directory server's schema to include the client profile object class and associated attributes.

Support of RBAC and Projects

Role-based access control (RBAC) and projects that were introduced in the Solaris 8 OE relied on a naming service to operate efficiently. Because the RFC2307 schema did not include object classes and attributes that supported these features, schema extensions are required if you want to use the features with the LDAP Client.

`nisdomain` Object

The Solaris 8 OE clients have the notion of belonging to a naming service domain. NIS uses this domain name to locate an NIS server that supports the domain the client belongs to. To provide an equivalent, the directory server maintains the name of the domain it serves. Here again, an object class and associated attribute is required, making a schema extension necessary.

`proxyagent` Account

The Solaris 8 OE access the naming service during the boot process. For example, host names might need to be resolved to IP addresses by applications started in the Solaris 8 OEs run command (`rc`) scripts. At boot time, a user is not logged in and the OE needs to bind to the directory server as someone. Instead of setting up anonymous access and lessening security, create an account that the client uses

during boot time and grant that account the appropriate permissions. Before a client can be initialized using this approach, the account must exist on the server. It must also be assigned a password for use during authentication.

`pam_unix` Support

The Solaris 8 OE provides two types of authentication methods that use LDAP as a naming service. Both of these are implemented as pluggable authentication modules (PAMs). The first, referred to as `pam_unix`, uses the traditional Solaris 8 OE method of authentication. This method stores passwords in a one-way hashing format called *crypt*. When a user logs in, the *crypt* representation of the password is retrieved from the naming service and compared with the password entered by the user after it is hashed using the same *crypt* algorithm. With this method, the authentication process takes place on the client and clear-text passwords are never sent over the network.

An issue with `pam_unix` support is that passwords stored on the server must be readable by someone so they can be retrieved. If the `pam_unix` style of authentication is deployed with LDAP as the naming service, a *proxyagent* account is used to retrieve the password. This means that this account must have the proper access rights to search for and retrieve passwords.

The other authentication method, called `pam_ldap`, performs the authentication on the directory server, so the password is never sent to the client in any form.

Self-Write Access Control

By default, directory servers assume that users should be able to modify the entry they use to bind to the directory. In most cases, this is a reasonable assumption. However, there are certain fields that a Solaris 8 OE user account entry contains that you would not want a user to modify. For example, you would not want users to change their `uidNumber` attribute to 0, effectively giving them root privileges. Therefore, the default access control instruction (ACI) requires modification.

Virtual List Controls

The virtual list view (VLV) is an extension for the LDAP v3 search operation. This extension is required to avoid overloading the client that makes a search request that returns a very large number of entries. For example, if you executed an `ldaplist passwd` command and had 10,000 users, 10,000 entries would be returned. Without browsing indexes, the maximum limit on the number of entries a server can return would most likely be exceeded.

Setting Up the Solaris 8 OE LDAP Client and Server Configuration

At the time the Solaris 8 OE was released, Netscape Directory Server 4.16.1 was the directory server version that Sun Microsystems, Inc. supported. This version of directory server software was co-packaged with the Solaris 8 OE media kit and could be downloaded from the Sun ONE/iPlanet Web site. In either case, it was contained in a single compressed tar file. The basic directory server installation steps were:

1. Uncompress and un-tar the file:
`directory-4.16-domestic-us.sparc-sun-solaris2.6.tar.gz`
2. Run the `setup` command from the installation directory.
3. Create an `rc` startup script to automatically start the directory server at boot time.

Once the directory server was installed, the `setup_native_LDAP.sh` script, which is available on the www.sun.com/blueprints Web site, was run to configure the server to support Solaris 8 OE LDAP Clients. The basic steps the script performed were:

1. Add the required schema definitions.
2. Create the directory information tree (DIT) structure.
3. Create the `nisDomain` object.
4. Set up appropriate access controls.
5. Generate an LDAP Client profile.
6. Create attribute indexes.
7. Create VLV indexes.

After the server was configured, you imported your NIS data by either running publicly available NIS migration scripts or by using the `dsimport` command, which was included in the NIS Extensions software.

When the directory server was configured and populated, you were able to initialize the Solaris 8 OE LDAP Clients. The easiest way to do this was by specifying the client profile that was created by the setup script on the command line when the `ldapclient` command is run. For example:

```
# ldapclient -P SolarisNative 192.9.200.1
```

One additional configuration step was required if you planned to use `pam_ldap` authentication instead of the default `pam_unix`. In this case, you also needed to modify the `/etc/pam.conf` file on the client. For example, you could replace the following lines:

```
login auth required /usr/lib/security/$ISA/pam_unix.so.1
```

with the following lines:

```
login auth sufficient /usr/lib/security/$ISA/pam_unix.so.1
login auth required /usr/lib/security/$ISA/pam_ldap.so.1
try_first_pass
```

Additional lines, not shown here, would also need to be modified to allow remote logins and password management functions.

Solaris 9 OE Secured LDAP Client and Server Configuration

The procedure for installing and configuring the directory server and client initialization has changed significantly in the Solaris 9 OE Secured LDAP Client implementation. The following sections explain the changes.

Directory Server Configuration

The most obvious change in the Solaris 9 OE is that the Sun ONE Directory Server 5.1 is the supported directory server in place of the Netscape Directory Server 4.1x.

In addition, the directory server software is packaged differently. Instead of being contained in a single compressed tar file, it is divided into several Solaris OE packages.

Note – The package format is only available with the Solaris 9 OE. For other versions of the Solaris OE, download the compressed tar file version.

The following 13 packages comprise the directory server software:

- IPLTadcon: Administration Server Console
- IPLTadman: Administration Server Documentation
- IPLTadmin: Administration Server
- IPLTcons: Console Client Base
- IPLTdscon: Directory Server Console
- IPLTdsman: Directory Server Documentation
- IPLTdsr: Directory Server (*root*)
- IPLTdsu: Directory Server (*usr*)
- IPLTjss: Network Security Services for Java™
- IPLTnls: Nationalization Languages and Localization Support
- IPLTnspr: Portable Runtime Interface
- IPLTnss: Network Security Services
- IPLTpldap: PerLDAP

▼ To Install the Solaris OE Packages

The Solaris OE packages are installed with the full distribution, or can be installed after the Solaris 9 OE is installed. If you install them later, you must install them in the correct sequence.

- **When you install the Solaris OE packages, install them in the correct sequence, as follows:**

```
PKGDIR=/cdrom/sol_9_sparc_2/Solaris_9/Product
pkgadd -d $PKGDIR IPLTdsman
pkgadd -d $PKGDIR IPLTadman
pkgadd -d $PKGDIR IPLTnspr
pkgadd -d $PKGDIR IPLTnls
pkgadd -d $PKGDIR IPLTnss
pkgadd -d $PKGDIR IPLTjss
pkgadd -d $PKGDIR IPLTpldap
pkgadd -d $PKGDIR IPLTcons
pkgadd -d $PKGDIR IPLTdscon
pkgadd -d $PKGDIR IPLTadcon
pkgadd -d $PKGDIR IPLTdsr
pkgadd -d $PKGDIR IPLTdsu
pkgadd -d $PKGDIR IPLTadmin
```

▼ To Install the Directory Server

A wrapper program called `/usr/sbin/directoryserver` is included in the Solaris 9 OE version of the directory server. The purpose of the wrapper is to simplify operations by removing path name dependencies. For example, the generic directory server installation allows you to specify the target directory and the instance name of the directory server you are installing. Therefore, you need to remember the path name where the directory server files were installed. The wrapper program figures out the correct path for you.

- To install the directory server, run `setup` from the wrapper, as follows:

```
# /usr/sbin/directoryserver setup
```

The dialogue that appears after you run `setup` is similar to what appears during a generic directory server installation. What is different is that you are not prompted to accept the server license, and the target directory and the directory server instance name are chosen for you. The target directory is `/usr/iplanet/ds5` and the instance name is `slapd-hostname`. While the files in these directories might look familiar to you, they should never be addressed directly. Instead, always use the wrapper program to perform operations on them.

Note – Never run the `uninstall` program directly. Use the `directoryserver` wrapper, instead. Failure to do so results in the deletion of critical files, which will prevent you from running `setup` again.

Netscape Directory Server 4.16.1 and Sun ONE Directory Server 5.1 Differences

The directory server supported with the Solaris 9 OE has many advantages over the previous version. However, different configuration procedures are required to set up the Solaris OE LDAP Client on the newer directory server because of these new features. This means that the `setup_native_LDAP.sh` script used to configure the Solaris 8 OE (see “Setting Up the Solaris 8 OE LDAP Client and Server Configuration” on page 8) will not work. The new features that affect configuration include:

Schema file storage—In the previous version, schema files were stored as `*.conf` files in an user-friendly format. The schema was separated into two files, one for object classes and one for attributes. New object classes and their attributes were placed in files called `slapd.user_oc.conf` and `slapd.user_at.conf`.

The new directory server stores schema files in LDAP Data Interchange Format (LDIF), which can contain object class and attribute definitions, in addition to the `slapd.user_oc.conf` and `slapd.user_at.conf` files. New definitions are added by creating a text file with an `.ldif` extension and placing it in the schema directory located under the server instance you are working with. By convention, the new schema definitions are placed in a file called `99user.ldif`. The files are processed in numeric order similar to the way Solaris OE `rc` scripts are processed.

Multiple database support—The previous version maintained all directory data in a single database, restricting the data to a single volume and allowing only one set of indexes for the whole directory. The new version allows multiple database backends that can reside on different volumes and have their own indexes. This means that some directory entities are referenced slightly different than in the previous version.

New authentication methods—The new directory server supports SASL/DIGEST-MD5 authentication that provides a method of encrypting a password before it is sent to the server. This feature first appeared in iPlanet Directory Server 5.0.

New LDAP Client Profile Format

To support the numerous Solaris 9 OE Secured LDAP Client enhancements, a new profile object class called `DUAconfigProfile` is defined. The following features are supported through attributes set in the new profile:

Service Search Descriptors—Allows you to specify a search path for naming service databases and filter the requests. For example, you can specify additional locations to search for database entries if an entry is not found in the first location. You can also assign an employee type such as *permanent* or *temporary* to account entries, then set up a filter to let only permanent employees log into a particular computer.

Attribute/Objectclass Mapping—Allows you to use a non-default object class or attribute. For example, defining user accounts somewhere other than the `posixAccount` object class.

Authentication Service Methods—Allows you to assign a different authentication method to a different service. For example, `pam_ldap` might use simple authentication with SSL/TLS, while the `passwd` command might use SASL/DIGEST-MD5.

The Solaris 9 OE Secured LDAP Client was designed to be backward-compatible with a server configured to support Solaris 8 OE LDAP Clients. The behavior of the client depends, in part, on the profile used initialized it.

Note – The Solaris 9 OE implementation requires the use of profiles and the LDAP cache manager.

The old profile type is identified as:

`NS_LDAP_FILE_VERSION = 1.0`

The new profile type is:

`NS_LDAP_FILE_VERSION = 2.0`

The profile type is determined by whether the profile contains the `DUAconfigProfile` object class or the `SolarisNamingProfile` object class. Profiles containing the former class are considered version 2.0.

New Automount Object Class

In the Solaris 8 OE implementation, `automount` maps were represented by the generic `nisObject` object class, which contained a key and an associated value. The Solaris 9 OE implementation introduces two new object classes to hold automap information: `automount` and `automountMap`.

You must take this difference into consideration when supporting both Solaris 8 OE and Solaris 9 OE LDAP clients on the same directory server.

New Security Features and Enhancements

The SSL/TLS libraries that are integrated with the Secured LDAP Client are an important addition to the Solaris 9 OE. This enhancement enables the client to run LDAP over the SSL/TLS transport, the benefit of which is that LDAP traffic can be encrypted to prevent eavesdropping.

Support of the SASL/DIGEST-MD5 authentication method is another security enhancement in the Solaris 9 OE. This method allows the encryption of just the password using a message digest. The details on how this actually works are very complex, but the configuration is not complicated.

The requirement that *proxyagent* credentials be stored in the client profile has been removed in the Solaris 9 OE. Instead of storing them in the profile, which is susceptible to snooping while the LDAP client cache is refreshed, they are maintained only in the local `/var/ldap/ldap_client_cred` file. This implies that Solaris 9 OE clients need to be supplied with the *proxyagent* credentials manually.

Command Differences

In the Solaris 9 OE, one new command has been added, one command has been dropped, and the syntax of another has changed. These changes are as follows:

`ldapaddent`—This new command can be used to populate the directory from text files in the `files` format. This command can only be run on a configured Solaris 9 OE Secured LDAP Client because the command determines what directory server to use and where to place the data in the DIT from the client profile.

If the client is configured to use a version 1 profile, automaps are mapped to the old style `nisObject` object class. If a version 2 profile is being used, they are mapped to the new `automount` and `automountMap` object classes.

`ldap_gen_profile`—This command has been replaced with the `genprofile` switch in the updated `ldapclient` command.

`ldapclient`—The syntax for this command has changed significantly. The switches such as `-l` and `-i` have been replaced with keywords like `list` and `init`. The new syntax appears as follows:

```
ldapclient [-v | -q] genprofile | init | manual | mod | list |
uninit [args]
```

where `args` takes the form of `-a attrName=attrVal`. For example, to specify a profile, you would use `-a profileName=myprofile`.

▼ To Configure the Sun ONE Directory Server 5.1 and the Solaris OE Secured Client

After installing and configuring the server with the `directoryserver setup` command, configure the server to support Secured LDAP Clients.

1. Run the `/usr/lib/ldap/idsconfig` script.

The script asks a number of questions, most of which have to do with setting up the parameters in the client profile.

Note – After the `idsconfig` script completes, you are instructed to run the `directoryserver vlvindex` command to create the VLV indexes. If you run the command before populating the directory server with data, error messages will be displayed. These messages are harmless and you can ignore them.

2. Initialize the client by running the following command:

```
ldapclient init -a profilename=myprofile 192.9.200.1 -a proxyDn -a proxypassword
```

3. If you are using `pam_ldap`, edit the `/etc/pam.conf` file.

▼ To Set Up SSL/TLS

To use SSL/TLS, both the server and clients need to be configured. The server must be set up to use port 636 for the encrypted channel, and port 389 for the clear-text channel. Before setting up the server you need to have the following items:

- A signed server certificate
- The signer's certificate

Obtain a service certificate from a certificate authenticator (CA) you set up or from a commercial one (**ID). Alternatively, use the `certutil` utility to create self-signed certificates. This certificate gets installed on the directory server. Once installed, SSL can be enabled.

Note – The server certificate is kept in a database that is password protected. You are prompted for this password each time the directory server starts, unless you place it in a special configuration file that is described in the Sun ONE Directory Server 5.1 documentation.

The client must have a certificate database that contains the signer's certificate. This database can be created by a Netscape browser, then moved to the appropriate place. See the Sun document, *System Administration Guide: Naming and Directory Services* manual, (part number 806-4077-10) for additional information.

▼ To Set Up the Sun ONE Directory Server 5.1 on the Solaris 8 OE

At the time this article was written, a Sun Microsystems, Inc. supported script that configured the Sun ONE Directory Server 5.1 running on the Solaris 8 OE was not available. The Solaris 9 OE `idsconfig` script will not work because of dependencies on the new and modified commands discussed earlier.

Most of the steps are the same as those used to configure the Netscape Directory Server 4.1x as described in the Sun BluePrints book, *Solaris and LDAP Naming Services*, with the following differences:

1. Update the schema files.

Obtain the `99user.ldif` file from a Solaris 9 OE directory server installation that is configured with the `idsconfig` command.

Note – If you are using the Sun ONE Directory Server 5.1, service pack 1, remove the reference to `automount` and `automountMap` in the `10rfc2307.ldif` file.

2. Create VLV indexes.

a. Import the following LDIF replacing `sun.com` with your domain name. For example, to create the VLV index for the `passwd` database:

```
dn: cn=sun.com_passwd_vlv_index, cn=user Root, cn=ldbm
database,cn=plugins, cn=config
objectClass: top
objectClass: vlvSearch
cn: sun.com_passwd_vlv_index
vlvBase: ou=people,dc=sun,dc=com
vlvScope: 1
vlvFilter: (objectClass=posixAccount)
aci:(target="ldap:///cn= sun.com_passwd _vlv _ index, cn=userRoot,
cn=ldbm database, cn=plu gins,cn=config")(targetattr="*")(version
3.0; acl "Config";allow(read,search,compare) userdn= "ldap:///
anyone";)

dn: cn=sun.com.getpwent,cn=sun.com_ passwd
_vlv_index,cn=userRoot,cn=ldbm database, cn= plugins,cn=config
cn: sun.com.getpwent
vlvSort: cn uid
objectClass: top
objectClass: vlvIndex
vlvEnabled: 1
vlvUses: 1
```

b. Repeat the preceding step, substituting the following attributes and object classes:

```
shadow- getspent, objectclass=ShadowAccount
group- getgrent,objectclass=PosixGroup
hosts - gethostent, objectclass=ipHost
networks - getnetent, objectclass=ipNetwork
rpc - getrpcent, objectclass=oncrpc
```

c. Run the following command from the *server-root* *server-instance* directory:

```
vlvindex -n userRoot -T sun.com.getpwent
vlvindex -n userRoot -T sun.com.getspent
vlvindex -n userRoot -T sun.com.getgrent
vlvindex -n userRoot -T sun.com.gethostent
vlvindex -n userRoot -T sun.com.getnetent
vlvindex -n userRoot -T sun.com.getrpcent
```

3. Create indexes.

Indexes are maintained for each database backend, so the `dn:` of the index needs to reflect this. For example:

```
dn: cn=uid,cn=index,cn=userRoot,cn=ldbm data base,cn=plugins,cn=config
objectClass=top
objectClass=nsIndex
cn=uid
nsSystemIndex=false
nsIndexType=eq
```

4. Create your client profiles with the `ldap_gen_profile` command.

Debugging Common Problems

After your directory server has been configured for LDAP Clients and populated with data, users should be able to log in. If any problems develop, it is helpful to understand the steps that are being performed when logging into a Solaris 8 or 9 OE system that is set up to use LDAP.

The `snoop` command can be a valuable tool in understanding these steps. The following example assumes that `pam_unix` and `pam_ldap` are stacked so that `pam_unix` is tried first. The credential level is *proxy*, and to support `pam_unix` logins with LDAP accounts, the *proxyagent* is granted password read permission.

To help illustrate the login process, encryption is not turned on. In a production environment, this would not be the case, so you would not be able to observe passwords in the clear.

▼ To Debug Common Problems

The first thing you should do if you encounter problems is to make sure the account is configured the way you think it should be.

1. To check the configuration of an account, run the `ldaplist` command on the client:

```
# ldaplist -l passwd tom
dn: uid=tom,ou=people,dc=sun,dc=com
objectClass: posixAccount
objectClass: shadowAccount
objectClass: account
objectClass: top
uid: tom
cn: Tom Bialaski
uidNumber: 12345
gidNumber: 10
gecos: Tom Bialaski - Enterprise Engineer
homeDirectory: /home/tom
loginShell: /bin/csh
userPassword:{SSHA}irwM5kVMuJgpYEd+S
shadowLastChange: 11892
shadowFlag: 0

From this listing, you can tell that the user tom does not have a
password stored in {crypt} format, so pam_ldap authentication
is used.

Next, turn on snoop on your directory server or client:
# snoop -v ldap | grep LDAP
```

From this listing, you can tell that the user `tom` does not have a password stored in `{crypt}` format, so `pam_ldap` authentication is used.

2. Next, turn on `snoop` on your directory server or client, as follows:

```
# snoop -v ldap | grep LDAP
```

The following example of a condensed `snoop` trace shows what takes place when user `tom` logs in. The bolded commands show what the example user types.

1. The user binds to the directory as the *proxyagent*, as follows:

```
Operation *[APPL 0: Bind Request]
[Object Name]
    cn=proxyagent,ou=profile,dc=sun,dc=com
Authentication: Simple [0]
    test1234
Operation *[APPL 1: Bind Response]
[Result Code]
    Success
```

Because, for this example, encryption is turned off, you can see the actual domain name and associated password the client is attempting to bind with. In this example, the bind operation is successful.

2. The user searches the directory to see if user `tom` exists, as follows:

```
Operation *[APPL 3: Search Request]
[Base Object]
    ou=people,dc=sun.com
Equality Match *[3]
[Attr Descr]
    objectClass
[Value]
    shadowAccount
    *[3]
[OctetString]
    uid
[OctetString]
    tom
```

If an entry with the `shadowAccount` object class and `uid=tom` is found, the operation is successful.

3. The user attempts to retrieve the user's password (remember that `pam_unix` is the first one in the stack), as follows:

```
Operation *[APPL 4: Search ResEntry]
[Object Name]
    uid=tom,ou=people,dc=sun,dc=com
*[Attribute]
    uid
[Value]
    tom
*[Attribute]
    userpassword
[Value]:
    {SSHA}irwM5kVMuJgpYEd+S
*[Attribute]
    shadowlastchange
[Value]
    11892
```

Because `pam_unix` expects to see the password returned in `{crypt}` format, the authentication fails and the next PAM in the stack is tried.

4. The user attempts to authenticate the password with `pam_ldap`, as follows.

```
Operation *[APPL 0: Bind Request]
[Object Name]
    uid=tom,ou=people,dc=sun,dc=com
Authentication: Simple [0]
    mysecret
```

If the bind succeeds, the user is authenticated.

5. The user retrieves user information, as follows:

```
Operation *[APPL 4: Search ResEntry]
[Object Name]
    uid=tom,ou=people,dc=sun,dc=com
*[Partial Attributes]
*[Attribute]
cn
[Value]
    Tom Bialaski
uid
[Value]
    tom
*[Attribute]
uidnumber
[Value]
    12345
*[Attribute]
gidnumber
[Value]
    10
*[Attribute]
gecos
[Value]
    Tom Bialaski - Enterprise Engineer
*[Attribute]
homedirectory
[Value]
    /home/tom
*[Attribute]
loginshell
    /bin/csh
```

The login information can now be used to set the home directory, execute a login shell, and set the user's user identifier (UID) and group identifier (GID).

6. The user locates the user's home directory, as follows:

```
Operation *[APPL 4: Search ResEntry]
  [Object Name]
automountKey=tom,automountMapName= au to_home,dc=sun,dc=com
objectClass
  [Value]
automount
  [Value]
top
*[Attribute]
automountKey
  [Value]
    tom
*[Attribute]
automountInformation
  [Value]
    nfssserver7:/export/home7/tom
```

The user is now logged in.

Note – The `.login`, `.cshrc`, and `.profile` files are examined and additional naming server lookups are performed. For brevity, these operations are not shown.

The Solaris 9 OE delivered the second phase of the LDAP Client that includes several security enhancements and is better integrated than the previous release. The Solaris 9 OE implementation is compatible with the Solaris 8 OE implementation so a single server can be configured to support both.

About the Author

Tom Bialaski is a Senior Staff Engineer in the Integrated Product Group division of Sun Microsystems. He is the author of *Solaris Guide for Windows NT Administrators* and a co-author of *Solaris and LDAP Naming Services*.

Acknowledgements

The author would like to recognize Michael Haines, Senior Staff Engineer for his contributions to this article.

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

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

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`