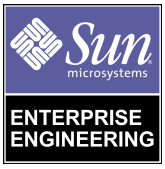
Using `dsimport` to Convert NIS Maps to LDAP Directory Entries

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ OnLine - February 2001



`http://www.sun.com/blueprints`

Sun Microsystems, Inc.
901 San Antonio Road
Palo Alto, CA 94303 USA
650 960-1300 fax 650 969-9131

Part No.: 806-7674-10
Revision 01, 01/29/01
Edition: February 2001

Copyright 2001 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, Sun BluePrints, iPlanet and Solaris are trademarks, registered trademarks, or service marks 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 United States 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.

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 2001 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, Sun BluePrints, iPlanet 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.

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

Using `dsimport` to Convert NIS Maps to LDAP Directory Entries

Replacing NIS servers with Lightweight Directory Access Protocol (LDAP) servers that support Solaris™ 8 Operating Environment (Solaris OE) native LDAP clients requires populating the directory with data contained in your existing NIS maps. Because the amount of data in the NIS maps can be quite large, some process automation will be required.

To automate the process, either run a conversion script that you or someone else has created, or run the `dsimport` utility. Because the `dsimport` method is supported by Sun, it is a best practice. This article focuses on the use of the `dsimport` utility and offers tips on how to convert your NIS data to LDAP directory entries.

Note – While using the `dsimport` utility to convert NIS data to LDAP directory entries you may encounter parsing problems. For example, some customers have experienced parsing problems due to the placement of hyphens in the GECOS field of the `passwd` map or use of special characters. In these cases, the NIS source files may need to be modified before they are used as input to `dsimport`.

Background of the *dsimport* utility

The dsimport utility was originally designed to support the Sun™ Directory Server to provide an LDAP to NIS synchronization service. Because this synchronization service is now included in the Solaris OE Extensions for iPlanet™ Directory Server, the dsimport utility is part of that package. The dsimport utility can be run automatically from the installation script when the NIS extensions are installed, or as a standalone utility independent of the Solaris OE Extension package. The focus of this article is on how to use the dsimport utility to populate an LDAP directory to support native LDAP clients. Because this is not the original purpose of the utility, modifications to the default configuration file, nis.mapping, are required.

How *dsimport* Works

The dsimport command is a SPARC™ executable that is run from the command line with several arguments. An input file is specified as an argument that is expected to be in the files format, for example, /etc/passwd. The output is an LDAP Data Interchange Format (LDIF) representation of NIS map entries. The following functions are performed by dsimport:

1. An NIS source file (files format) is read.
2. The file is parsed to extract the relevant LDAP attributes and to determine which LDAP object classes to place them in.
3. An ldapsearch operation is performed on the directory to see if some of the parsed information already exists in the LDAP directory.
4. Based on the result of the ldapsearch operations, LDIF, which represents new or updated data, is created.
5. The LDIF is imported into the directory, creating new or updated entries.

The -n switch causes dsimport to stop after Step 4, so you can examine the LDIF output file for errors before it is imported. Unless you are sure there are no errors, the -n switch should be used, followed by a manual import of the LDIF file using the ldapmodify command.

It is recommended that the NIS source files used to create your NIS maps be specified as the input files, although they could be generated with the ypcat command. For automount maps and custom NIS maps, the -k option of ypcat needs to be specified to generate the correct output files. Using the actual NIS source files avoids this issue.

The parsing is performed based on rules established in the `nis.mapping` file, which can be modified to fit your directory structure. Because the default `nis.mapping` file is designed to work with the NIS extensions and not native LDAP, it needs to be modified as discussed in the following sections.

How Data is Converted

The `nis.mapping` file contains a set of rules for how data is to be placed in the LDAP directory. The default `nis.mapping` file is located in `/opt/SUNWconn/ldap/default/mapping` directory. The syntax of the file is rather cryptic and is explained in detail in the `NIS.pdf` document which is part of the Solaris OE Extensions for the iPlanet Directory Server. The PDF file can be found in the <install-dir>**/Solaris_Extension/Doc** directory of the extension package which can be downloaded from:

`www.iplanet.com`

Because the default `nis.mapping` file assumes you are running the NIS extensions, several modifications need to be made. A template which includes these modifications can be downloaded from the `sun.com/blueprints` web site. Only minor modifications are then required to configure the template for your specific environment.

Changes to the *`nis.mapping`* File

Complete details of the nis.mapping file are contained in the `NIS.pdf` document, which should be read before performing any modifications. A condensed version is presented in this section.

The `nis.mapping` file contains a front-end section, and sections pertaining to the individual NIS maps. The front-end section contains a couple of variables that need to be changed (as shown in the following examples):

```
Common:

   ########################################
   # configuration variables          #
   ########################################

   # The name of the NIS domain
   DOMAIN_NAME=airius.com

   #
   # NAMING_CONTEXT, if defined, gives the root of the naming tree
   # if it is not defined, the naming tree root is derived from
   # the DOMAIN_NAME variable using dc attributes for each
   # element in the domain name (airius.com --> dc=airius,dc=com)
   # NAMING_CONTEXT=O=XYZ,C=US
   #
   # Where to create the LDAP administrative entries associated with
   the maps.
   ADMIN_SUFFIX=ou=admin,ou=Services,o=airius,c=us

   # directory where NIS binaries maps are generated
   DBM_DIRECTORY=/var/yp

   # Specifies whether to push changes to NIS maps automatically,
   # and the delay (in minutes)
   AUTOMATIC_PUSH=disabled
   AUTOMATIC_PUSH_DELAY=5
```

The modified version for native LDAP looks as follows:

```
Common:

   ########################################
   # configuration variables          #
   ########################################

   # The name of the NIS domain
   # DOMAIN_NAME=sun.com
   DOMAIN_NAME= blueprints.com

   #
   # NAMING_CONTEXT, if defined, gives the root of the naming tree
   # if it is not defined, the naming tree root is derived from
   # the DOMAIN_NAME variable using dc attributes for each
   # element in the domain name (airius.com --> dc=airius,dc=com)
   # NAMING_CONTEXT=O=XYZ,C=US
   #

  # Where to create the LDAP administrative entries associated with
    the maps.
   ADMIN_SUFFIX="cn=Directory Manager"

   # directory where NIS binaries maps are generated
   DBM_DIRECTORY=/var/yp

   # Specifies whether to push changes to NIS maps automatically,
   # and the delay (in minutes)
   AUTOMATIC_PUSH=disabled
   AUTOMATIC_PUSH_DELAY=5
```

The DOMAIN_NAME variable should be changed to reflect your own domain name. In this article, the fictitious blueprints.com domain is used in the examples. The ADMIN_SUFFIX variable is changed because the layout of the tree used by native LDAP does not include the ou=admin,ou=Services branch. The lines below the ADMIN_SUFFIX contain information relevant to the NIS extensions, and therefore have no effect with native LDAP.

The next portion of the nis.mapping file shows what an NIS map (or table entry) looked like in the original file. For brevity, only the Common and Dynamic sections are shown. The table chosen in this example is the hosts table.

```
  ########################################
  ## hosts                              ##
  ########################################

  Table: hosts.byaddr

    Common:
      MAP_NAME=hosts.byaddr
      PRIVATE_OBJECTCLASSES=ipHost

    Dynamic:
      LINE =>$ipHostNumberT $remainderT
      ipHostNumberT=$NIS_KEY

     MATCH_FILTER=(&(objectClass=ipHost)(ipHostNumber=$ipHostNumber
     T))
      ALL_FILTER=(&(objectClass=ipHost)(ipHostNumber=*))
      DC_NAMING=split($DOMAIN_NAME, ".", "dc=", ",", left2right)
     rootTreeT=ou=Services,$NAMING_CONTEXT||ou=Services,$DC_NAMING
      BASE_DN=ou=Hosts,$rootTreeT

...

Table: hosts.byname

    Common:
      MAP_NAME=hosts.byname

    Dynamic:
      ipHostNameT=$NIS_KEY
      MATCH_FILTER=(&(objectClass=ipHost)(cn=$ipHostNameT))
      ALL_FILTER=(&(objectClass=ipHost)(cn=*))
      DC_NAMING=split($DOMAIN_NAME, ".", "dc=", ",", left2right)
      rootTreeT=ou=Services,$NAMING_CONTEXT||ou=Services,$DC_NAMING
      BASE_DN=ou=Hosts,$rootTreeT
```

The following shows what the native LDAP version should look like:

```
########################################
## hosts                              ##
########################################

Table: hosts

  Common:
    MAP_NAME=hosts
    PRIVATE_OBJECTCLASSES=ipHost

  Dynamic:
    LINE =>$ipHostNumberT $remainderT
    ipHostNumberT=$NIS_KEY
     MATCH_FILTER=(&(objectClass=ipHost)(ipHostNumber=$ipHostNumberT))
    ALL_FILTER=(&(objectClass=ipHost)(ipHostNumber=*))
    DC_NAMING=split($DOMAIN_NAME, ".", "dc=", ",", left2right)
    rootTreeT=ou=Hosts,$NAMING_CONTEXT||ou=Hosts,$DC_NAMING
    BASE_DN=$rootTreeT
```

The main difference between these two is that in the original file the `hosts` section was split into two tables: the `hosts.byaddr` and `hosts.byname`. This was done to support the NIS to LDAP synchronization service—although it is not required for native LDAP. The Export section (not shown) has been removed from the native LDAP version because it is used to generate NIS source files from LDAP entries and is not required to support native LDAP. There are also a few minor changes to each table section.

The table entries assume default containers of `ou=hosts`, `ou=services`. These are the containers that the native LDAP client assumes by default, so it is not a recommended practice to change them. Although there are vast changes to the `nis.mapping` file to support native LDAP, the template files includes them. You should only have to modify one or more variables in the front-end section.

Mapping NIS Maps to Object Classes and Attributes

Each table entry in the `nis.mapping` file contains an import section. The import section is a set of rules which define how the data in the input file is to be interpreted, and how corresponding LDAP entries are created. The format of LDAP

entries is defined by an object class, which is a template for what type of data is to be stored. There is not a one to one mapping of NIS map entries to LDAP entries. For example, entries of the object class `ipHost` contain data from multiple NIS maps, and also (as is the case with the `publickey` map), the data is contained in more than one entry.

To understand how data items get mapped, the following table is provided. The first column lists the database name as it appears in the `/etc/nsswitch.conf` file, the second column lists the object class (or classes) where the data is stored, and the third column lists the containers which should be used to place the LDAP entries into.

TABLE 1 NIS Map Object Classes and Containers

nsswitch Database	Objectclass	Recommended Container
passwd	posixAccount, shadowAccount	ou=people
group	posixGroup	ou=group
hosts	ipHost	ou=hosts
ipnodes	ipHost	ou=hosts
ethers	ieee802Device	ou=hosts
netmasks	ipNetwork	ou=networks
networks	ipNetwork	ou=networks
bootparams	bootableDevice	ou=hosts
rpc	oncRpc	ou=rpc
services	ipServices	ou=services
protocols	ipProtocol	ou=protocols
netgroup	nisNetwork	ou=netgroup
aliases	mailGroup	ou=aliases
automount	nisObject	nismap-name=auto_*
publickey	nisKeyObject	ou=people, ou=hosts
auth_attr	SolarisAuthAttr	ou=SolarisAuthAttr
exec_attr	SolarisExecAttr	ou=SolarisExecAttr
prof_attr	SolarisProfAttr	ou=SolarisProfAttr

As shown in this table, the `automount` database resides in more than one container. The entries in the ou=hosts container have data taken from the `hosts`, `ipnodes`, `ethers`, `bootparam` and `publickey` databases.

User Accounts

As shown in the previous table, the LDAP user account entries in `ou=people` contains data from two sources: `/etc/passwd` and `/etc/shadow`. This data is stored in two object classes: `PosixAccount` and `shadowAccount`. Both of these object classes are auxiliary ones, which means that they must be added to a structural object class. The default structural object class which `dsimport` uses is the `account` object class.

The following LDAP attributes are defined in the `posixAccount` object class.

TABLE 2 The `posixAccount` Attributes

Attribute	Description
cn	Name of the person (common name)
uid	User's login name
uidNumber	User's numeric ID - the Unix UID
gidNumber	User's numeric group ID - the Unix GID
homeDirectory	Absolute path to the user's home directory
userPassword	User's password
loginShell	User's login shell
gecos	The Unix comment field
description	A description of the user

Notice how the attributes are aligned with the values found in `/etc/passwd` entries with the exception of `description`, which is a commonly used LDAP attribute that is not referenced by the name service switch. The `cn` attribute is derived from the first two words in the `gecos` field because it is common practice to enter the user's first name and surname in this field. Also note that the `uid` attribute refers to the user's login name and not the numeric UID.

The following LDAP attributes are described in the `shadowAccount` object class:

TABLE 3 `shadowAccount` Attributes

Attribute	Description
uid	User's login name
userPassword	User's password
shadowLastChange	Number of days since last password change
shadowMin	Minimum number of days required between password changes
shadowMax	Maximum number of days required between password changes
shadowWarning	Number of days for warning of password expiration
shadowInactive	Number of days inactivity is allowed
shadowExpire	Date on which the user's login will be disabled
shadowFlag	Reserved attribute - not yet implemented
description	Description of user

The `posixAccount` and `shadowAccount` object classes share three attributes: `uid`, `userPassword`, and `description`. The `uid` attribute is usually populated from `/etc/passwd`; the `userPassword` attribute is populated from `/etc/shadow`. The `description` attribute is optional and is populated from in-line comments. The remaining attributes pertain to password aging. The values from `/etc/shadow` are entered here (if they exist).

Note – Password aging using these attributes is not implemented in native Solaris OE LDAP. The iPlanet Directory Server provides its own mechanism for password aging and is used instead.

Running the dsimport Utility

The dsimport utility is part of the Solaris OE Extensions for iPlanet Directory Server software which can be downloaded from the iPlanet website at:

www.iplanet.com

The software comes in a tarfile called sol_ext41.tar. To extract the dsimport utility from the tarfile enter the following commands:

```
# tar xvf sol_ext41.tar
# cd Solaris_Extension
# pkgadd -d . SUNWdsutl
```

The dsimport utility is installed in the /opt/SUNWconn/ldap/sbin directory. Documentation on this utility is found in NIS.pdf under the Solaris_Extension/Doc directory. The default mapping file, called nis.mapping, is found in /opt/SUNWconn/ldap/default/mapping. As noted earlier, this file should be replaced with the one available from the www.sun.com/blueprints website.

There are many switches supported by dsimport utility, which are described fully in the NIS.pdf document. However, the examples in this article use only the ones listed below:

dsimport [-h *ldap_host*] [-m *mapping file*] [-t NIS *table*][-D *bindDN*] [-w *password*] filename, where:

-h The name of the LDAP server you are importing the NIS data to. The default is the local system.

-m The name of the nis.mapping file to use. The default is /opt/SUNWconn/ldap/default/mapping/nis.mapping.

-t The name of the NIS table you are importing. This tag must appear in the nis.mapping file you are using.

-D The distinguished name (DN) you are binding to the directory with.

-w The password associated with the DN you are binding as.

filename The name of the file in /etc format that you are importing.

Examples of how the dsimport utility is used to import the various NIS maps are provided in the following section.

Importing *passwd* and *shadow*

Creating accounts for users to log in requires information that is maintained in the `passwd.byname` map. This map is generated from two files; *passwd* and *shadow*. Although the `/etc/passwd` and `/etc/shadow` can be used as the input file, most system administrators maintain copies of these files in a different directory for security reasons. You should use the same files that are used to generate the `passwd.byname` map as the input file for `dsimport`.

An alternative way to generate the `passwd` file is to run the `ypcat` or `getent` command. However, if NIS is set up to hide the password (which is recommended for security), you will not be able to import user passwords unless you also import the `shadow` file.

The syntax for importing the `passwd` and `shadow` maps is as follows:

```
# dsimport -n -m nis.mapping -t passwd -D “cn=Directory Manager” \
-w mysecret passwd > passwd.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
passwd.ldif
# dsimport -n -m nis.mapping -t shadow -D “cn=Directory Manager” \
-w mysecret shadow > shadow.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
shadow.ldif
```

Note – The import of `shadow` is still required even if you are using a `passwd` file that contains passwords. This is because the `shadowAccount` object class must be added to the entry even though the `posixAccount` object class also contains the `userPassword` attribute. If you do not have a separate shadow file, you can run `dsimport` using the `passwd` file specifying the `-t shadow` option.

Importing *hosts*

As shown in Table 1, entries in the `ou=hosts` container has information from several NIS maps such as `ethers` and `bootparams`. The information in these other maps is stored in attributes contained in auxiliary object classes that get added to the `ipHost` object class, therefore, you should import `hosts` before the other associated NIS maps.

Before importing the `hosts` file, remove any blank or comment lines. These will confuse the `dsimport` parser. In-line comments can be left in as the information in them will be placed in the `description` attribute. The following example shows how to import the hosts and associated NIS maps:

```
# dsimport -n -m nis.mapping -t hosts -D “cn=Directory Manager” \
-w mysecret hosts > hosts.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
hosts.ldif
# dsimport -n -m nis.mapping -t ethers -D “cn=Directory Manager” \
-w mysecret ethers > ethers.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
ethers.ldif
# dsimport -n -m nis.mapping -t bootparams -D “cn=Directory \
Manager” -w mysecret bootparams > bootparams.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
bootparams.ldif
```

Importing Automount Maps

The NIS maps used by the automounter are somewhat different than other standard NIS maps. Instead of a single map, there are several maps which contain automounter data that quite often include customer defined maps. At a minimum, you probably have the `auto_master`, `auto_direct`, and `auto_home` maps. Because the content of each map is kept in a separate LDAP container, one needs to be created for each automount map. The automount table is specified for each map, with the `-V` option used to change the value of `MAP_NAME`.

The following examples show the creation of entries for `auto_master`, `auto_direct`, and `auto_home`, but you can easily substitute the name of any custom maps you have defined:

```
# dsimport -n -m nis.mapping -t automount -V \
MAP_NAME=auto_master \ -D “cn=Directory Manager” -w mysecret \
auto_master > auto_master.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
auto_master.ldif
# dsimport -n -m nis.mapping -t automount -V \
MAP_NAME=auto_direct -D “cn=Directory Manager” -w mysecret \
auto_direct > auto_direct.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
auto_direct.ldif
# dsimport -n -m nis.mapping -t automount -V \
MAP_NAME=auto_home -D “cn=Directory \ Manager” -w mysecret \
auto_home > auto_home.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
auto_home.ldif
```

Note – If you examine the LDIF file for `auto_master`, you may see encoded data in the `nismapentry` attribute. This is because of unprintable characters contained the `auto_master` input file. However, the data will be input correctly into LDAP.

Importing other Standard NIS Maps

The other standard NIS maps include:

- group
- services
- protocols
- rpc
- networks
- netmasks
- netgroup
- alias
- publickey

The data in these maps is inputted in a similar fashion to `hosts`, except another table name with the `-t` option is specified along with a different input file name. For example, adding the `group` map looks as follows:

```
# dsimport -n -m nis.mapping -t group -D “cn=Directory Manager” \
-w mysecret group > group.ldif
# ldapmodify -a -D “cn=Directory Manager” -w mysecret -c -f \
group.ldif
```

Note – The information in the `publickey` map is added to entries in `ou=people` and `ou=hosts`, therefore, these entries should be created before importing that map.

Verification and Troubleshooting

As a best practice, you should perform the following steps while importing data:

1. Check the ASCII source file and remove any blank or comment lines.
2. Examine the LDIF file created by `dsimport` for any irregularities.
3. After running `ldapmodify`, check the directory server’s error log.
4. Perform a `ldapsearch` on the directory to view the entries.
5. Run the `gentent` command to make sure the name service switch is working correctly.

The following is an abbreviated example of what the LDAP output file from `dsimport` looks like for the `passwd` map:

```
# cat passwd.ldif
dn: cn=Troy Brown,ou=People,dc=blueprints,dc=com
changetype: add
uid: tbrown
userpassword: {crypt}x
uidnumber: 12000
gidnumber: 10
gecos: Troy Brown - WR
homedirectory: /home/tbrown
loginshell: /bin/csh
objectclass: top
objectclass: account
objectclass: posixAccount
........
```

The following is the abbreviated output of `ldapsearch`:

```
# ldapsearch -D “cn=directory manager” -w mysecret -b ou=people, \
dc=blueprints,dc=com objectclass=posixaccount
dn: uid=tbrown,ou=People, dc=blueprints,dc=com
cn: Troy Brown
uid: tbrown
givenname: Troy
sn: Brown
uidnumber: 12000
objectclass: top
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: shadowaccount
objectclass: posixaccount
gidnumber: 10
homedirectory: /home/tbrown
gecos: Troy Brown - WR
loginshell: /bin/sh
userpassword: {crypt}MCN0TZTir.vTY
.......
```

Note – The preceding example assumes you have imported the `shadow` map so the password will be displayed. If it was not imported an “x” appears.

The following example is run from the Solaris OE LDAP client to display an entry:

```
# getent passwd tbrown
tbrown::12000:10:Troy Brown - WR:/home/tbrown:/bin/sh
#
```

If the `ldapsearch` command works, but the `getent` command does not, you may have a permission problem. Check the access log for possible errors.

Conclusion

This article described a method to import your NIS maps into an LDAP directory using the `dsimport` utility. Although this tool was originally designed to support a NIS to LDAP synchronization service, it works quite well with the native LDAP implementation after some modifications to the default configuration file. Use of this tool rather than a homegrown one or one found in the public domain is considered a best practice because it is complete, tested, verifiable, and most importantly supported by Sun.

Author's Bio: Tom Bialaski

Tom Bialaski is currently a Senior Staff Engineer with the Enterprise Engineering group at Sun Microsystems, and is the author of "Solaris Guide for Windows NT Administrators," and co-author of "Solaris and LDAP Naming Services." Tom has nearly 20 years of experience with the UNIX® operating system and has been a Sun Engineer since 1984.