

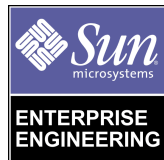


# Demystifying the LDAP Directory Information Tree (DIT)

---

*By Tom Bialaski - Enterprise Engineering*

*Sun BluePrints™ OnLine - April 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.: 816-0713-10  
Revision 01, 04/10/01  
Edition: April 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, and Solaris are trademarks, registered trademarks, or service marks of Sun Microsystems, Inc. in the U.S. and other countries.

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, 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 REPOUDRE 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

# Demystifying the LDAP Directory Information Tree (DIT)

---

Migrating from a legacy Solaris™ Operating Environment (Solaris OE) naming service, like NIS, to an LDAP based one requires moving your naming service data from a Solaris OE file-system to an LDAP Directory Information Tree (DIT). Understanding how data is stored in the DIT is key to a successful migration. Before you can map out how your data will be stored in the DIT, you need to understand some basic LDAP terms and concepts.

Both LDAP directories and file-systems, like UNIX® UFS, have a well-defined structure for placing data in them. While both store data in a tree structure, they have differences between them that significantly affect how and where you store data. Some of these differences are subtle but can cause confusion about how to organize your data if they are not well understood. This confusion has led to configuration problems encountered while setting up directory servers to support native LDAP in the Solaris OE.

Since all readers have some experience interacting with file-systems, this article draws an analogy to them that explains how the LDAP DIT is defined in general and how to correctly place data into it. This article presents examples that represent actual steps performed during configuration of a Netscape™ Directory Server (Netscape DS) to support Solaris OE native LDAP. In addition, this article discusses how to perform these steps and recommends DIT topologies to guide the reader through the planning phase of a NIS to LDAP migration.

---

## LDAP Terminology

Before explaining the differences between a file-system structure and an LDAP directory, you need to be familiar with some LDAP terminology. This section describes the following LDAP-related terms that are used throughout this article.

- *Entry* – A basic storage element that stores all LDAP items.
- *Container* – An element that stores entries or other containers.
- *DN (distinguished name)* – A name that uniquely identifies entries and is similar to a file-system path-name.
- *Suffix* – A DN that represents the root level of a directory or database back-end.
- *DSA (directory system agent)* – A term used in the X.500 specification which equates to a directory server.
- *DSE (directory server entry)* – A one-per-server entry that contains information relevant to the entire directory server, such as its schema.
- *Naming Context* – An entry in the DSE that specifies a large collection of entries.
- *Search Base* – A branch in the tree that is a starting point for a search operation.
- *Scope* – Specifies how many levels to search below the specified search base.

*Entries* are instances of objects defined in the directory's *schema*. The schema is a set of rules that define object classes and the attributes they contain along with what type of data can be stored. Object classes act as templates for creating entries. Every item that is stored in an LDAP directory is an entry. In a general sense, entries are like files and directories that make up the basic elements of a file system. However, directory entries are much more complex in structure.

*Containers* are regular entries that create a branch point in the DIT. Any entry can be used as a container, although organization unit (ou) objects are commonly used. The container does not maintain a list of entries; rather, it only serves as a component of a DN that specifies part of a path in a search base.

*DNs* are unique identifiers within a directory that is composed of several components. These components consist of an attribute/value pair, container names, and a directory suffix. The attribute chosen can be any attribute as long as its value is unique within the subtree where it resides. Following the attribute, container names may appear with the suffix as the last component. The exception to this is the root entry DN which consists of an attribute of o= or dc=. The DN is used to identify entries during bind operations (authentication) and when entries are created, deleted, and modified. The first component of a DN is an attribute that is contained in the entry that the DN references.

*Suffixes* are listed in the DSE when it is created. An entry is associated with a suffix that is commonly referred to as the root entry for the DIT. The last component of a DN is always a directory suffix. There can be, and usually are, multiple suffixes in a directory. To illustrate what a suffix is, suppose you have the following two entries:

- dc=bpsrus, dc=com – in the database back-end
- ou=People, dc=bpsrus, dc=com – also in the database back-end

Then, `dc=bpsrus, dc=com` is the suffix, and `ou=People, dc=bpsrus, dc=com` is a subtree, which is also a container used to store other entries.

*DSEs* are single entries that are created when the directory server is installed. They contain server-wide configuration information including the versions of the LDAP protocol that are supported; a list of server controls; extended operations and SASL mechanisms supported by the directory server; the URL's of alternative directory servers to contact if the primary directory server is unavailable; and the naming contexts or suffixes. The naming contexts or suffixes include the portion of the directory tree that is managed by this particular directory server.

*Naming Contexts* are defined as top level entries with no parent but with a database back-end. The following example illustrates what a naming context is.

Suppose you have the following entries in your directory:

- a) `dc=bpsrus, dc=com` – back-end for `bpsrus`.
- b) `dc=bp, dc=bpsrus, dc=com` – back-end for `bp.bpsrus`
- c) `ou=People, dc=bpsrus, dc=com` – back-end for `bpsrus`
- d) `ou=People, dc=bp, dc=bpsrus, dc=com` – back-end for `bp.bpsrus`

The following statements about these entries are true:

- a) - Is a naming context, a suffix, and a subtree
- b) - Is a suffix and a subtree, but not a naming context
- c) - Is a subtree only
- d) - Is a subtree only

The component `dc=com` is not an entry, and it is not a naming context because it does not have an associated database.

*Search Bases* refer to a portion of a DN or subtree. The last component of the search base is a suffix. Search bases are used as a starting point for LDAP searches and can be used in conjunction with a scope declaration to limit the number of levels searched.

---

## Key Differences

This section identifies the three key differences between file-systems and LDAP directories that you should be aware of. These differences are:

- File-systems have a root (/) which is the entry point into other directories. Conceptually the LDAP DIT has a root entry; however, only information about the directory server is stored there.
- File-system directories do not store data, LDAP containers can.
- File-system path-names are specified root to leaf node, LDAP DN's are specified leaf node to root.

The following sections explain these differences in greater detail.

## The Directory Root

Each suffix within a directory identifies the top node of a DIT. However, unlike a file-system that has an absolute path-name which always starts at “/”, you need to know the name of the top node or suffix of the tree you want to search. The name of suffixes can be determined by examining the root DSE. To do this, use the following command.

```
blueprints# ldapsearch -s base -b "" objectclass=*
dn:
objectclass: top
namingcontexts: dc=bpsrus,dc=com
namingcontexts: o=NetscapeRoot
subschemasubentry: cn=schema
supportedcontrol: 2.16.840.1.113730.3.4.2
supportedcontrol: 2.16.840.1.113730.3.4.3
supportedcontrol: 2.16.840.1.113730.3.4.4
supportedcontrol: 2.16.840.1.113730.3.4.5
supportedcontrol: 1.2.840.113556.1.4.473
supportedcontrol: 2.16.840.1.113730.3.4.9
supportedcontrol: 2.16.840.1.113730.3.4.12
supporteddsaslmechanisms: EXTERNAL
supportedldapversion: 2
supportedldapversion: 3
dataversion: blueprints.blueprints.com:389 020010123205331
netscapemdsuffix: cn=ldap://:389,dc=server1,dc=bpsrus,dc=com
vlvsearch: cn=getspent,cn=config,cn=ldb
vlvsearch: cn=getnetent,cn=config,cn=ldb
vlvsearch: cn=gethostent,cn=config,cn=ldb
vlvsearch: cn=getgrent,cn=config,cn=ldb
```

The `ldapsearch` command in this example uses the `-s` argument to specify base as the scope of the search. This results in only the content of the root DSE being returned. As you can see from the output, a root DSE contains information about the LDAP controls that the server supports; the LDAP versions supported; and the

Virtual List View indexes that are available. LDAP clients connecting to the server use this information to determine how to access data, but it is not relevant to the article; only the lines in bold print are discussed.

As shown, there are two `namingcontexts` lines that identify directory suffixes. The `dc=bpsrus,dc=com` suffix was specified in response to a prompt when the installation script was run. The `o=NetscapeRoot` suffix was created automatically when Netscape DS was installed to store configuration data which the server requires when it starts. This suffix is maintained by the server and should never be used to store other directory data. The `netscapemdsuffix` line identifies the instance of the Netscape DS running on the system. In this case, Netscape DS is running on the default port 389 on the server of which the DNS address is `server1.bpsrus.com`. Notice that two different naming conventions, `dc=` and `o=`, are used to identify the suffix. The next section describes what these are.

To find the starting point of a DIT, you need to query the root DSE to see what suffixes are present unless you already know what the suffix is. The reason why you need to query the root DSE is primarily because the LDAP protocol is a silent protocol, and thus does not advertise itself. Many applications, including the Solaris OE LDAP client, do this for you.

The following table summarizes all the information available in the root DSE entry.

Attribute Name	Description of Values
<code>namingContexts</code>	The values of this attribute are the naming contexts supported by this server (for example, "dc=bpsrus,dc=com").
<code>altServer</code>	The values of this attribute are LDAP URLs that identify other servers that can be contacted if this server is unavailable.
<code>supportedExtension</code>	The values of this attribute are the object identifiers (OIDs) of the LDAP v3 extended operations supported by this server. If this attribute is not in the root DSE, the server does not support any extended operations.
<code>supportedControl</code>	The values of this attribute are the object identifiers (OIDs) of the LDAP v3 controls supported by this server. If this attribute is not in the root DSE, the server does not support any LDAP v3 controls.

Attribute Name	Description of Values
supportedSASLMechanisms	The values of this attribute are the names of the SASL mechanisms supported by the server. If this attribute is not in the root DSE, the server does not support any SASL mechanisms.
supportedLDAPVersion	The values of this attribute are the versions of the LDAP protocol supported by this server (for example, 2 and 3).

TABLE 1 Information Available in the Root DSE Entry

## Suffix Naming Conventions

Two descriptors are commonly used as naming conventions to denote directory suffixes. These descriptors are domain component (dc) and organization (o) and either one can be used. The dc= notation is becoming more popular, while the o= notation dates back to X.500 which precedes LDAP. In some cases, the notation may have already been established by corporate IT architects, so you may not have a choice of which one to use.

If the dc= notation is chosen, a series of domain components is typically used which is aligned with your DNS address. For example, if you have a DNS domain called east.bpsrus.com, then the domain components are listed as dc=east,dc=bpsrus,dc=com. While this is usually tied to your DNS domain or subdomain, it is not required to be.

The attribute organization (o=) is a string that is specified in a single descriptor. The string can contain dots and can be expressed as: o=east.bpsrus.com. Either naming convention works fine. The only consideration is that the directory enabled applications you are running must be able to support it. You should choose one convention throughout your enterprise, then stick to it.

Examples of three naming contexts supported are:

1. X.500 naming

o=sun, c=US

2. DNS alignment

o=sun.com

3. The domain component

dc=sun, dc=com

## Creating Suffixes

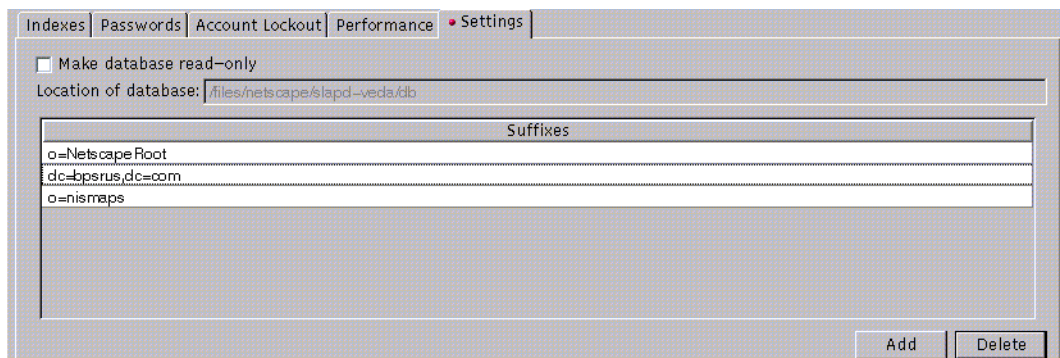
A main directory suffix is automatically created during the installation with the name supplied when prompted for a directory suffix. For example:

```
Suffix[o=server1.bpsrus.com]
```

The default suffix that appears during the Netscape DS 4.x installation (o=server1.bpsrus.com) specifies an organization (o=) object set to the fully-qualified name of the server it is being installed on. You can change the default during the installation by specifying a different suffix. For example:

```
Suffix[o=server1.bpsrus.com] dc=bpsrus,dc=com
```

You can create a suffix after the directory is installed. The easiest way to do this is through the Directory Console; go to the **Configuration->Database->Settings** tab. FIGURE 1 illustrates the Netscape Directory Console screen.



**FIGURE 1** Netscape Directory Console Screen

To add a suffix, simply press the **Add** button and input the suffix name on the blank line that appears. The change takes effect when you press the **Save** button (not shown here).

As an alternative to using the Directory Console for creating suffixes you can import LDIF that looks similar to this:

```
dn: cn=config, cn=ldbm
changetype: modify
add: nsslapd-suffix
nsslapd-suffix: o=newsuffix
```

However, there is one major difference between a suffix that you create during installation and one that you add after. Creating a suffix after the installation only creates an entry in the DSE and does not create a root entry. You must create one before you can populate the tree. This is why no object appears in the Directory Console window even though a new suffix is added.

You can create a root entry by importing an LDAP Directory Interchange Format (LDIF) file. For example, suppose you add a suffix called `o=nismaps` to an existing server and want to create a root entry for it. The LDIF file to do this contains the following lines:

```
dn: o=nismaps
objectclass: top
objectclass: organization
o: nismaps
```

This LDIF creates the root entry of a DIT that is addressed with a search base of `o=nismaps`. However, you do not necessarily need to create a new suffix if all you want to do is add a subtree under an existing DIT. The next section describes how to do this.

## Creating a Subtree

You can create a subtree or branch point by placing a container under the suffix or under another container. For example, you can create a subtree under the suffix `dc=bpsrus,dc=com` with the following LDIF:

```
dn: o=nismaps,dc=bpsrus,dc=com
objectclass: top
objectclass: organization
o: nismaps
```

In this example, the `dc=bpsrus,dc=com` suffix already exists and `o=nismaps` is added later by importing an LDIF file to create a branch. Besides creating a new suffix or branch point, additional information is required before NIS map data can be stored. The next section discusses options for creating a DIT for NIS map data storage.

---

## Planning the NIS Map Tree

You have several options for creating a repository for storing NIS map data. You can place map data directly under the suffix that was created when the directory server was installed; you can create a new suffix; or you can create a subtree under an existing suffix. Which method you choose may depend on your company directory naming policy. Whichever method is deployed, the top entry needs to contain a `nisDomainObject` object that LDAP clients can query to determine which NIS domain the server supports. The procedure for adding this object appears later in the article.

## NIS Data Storage Options

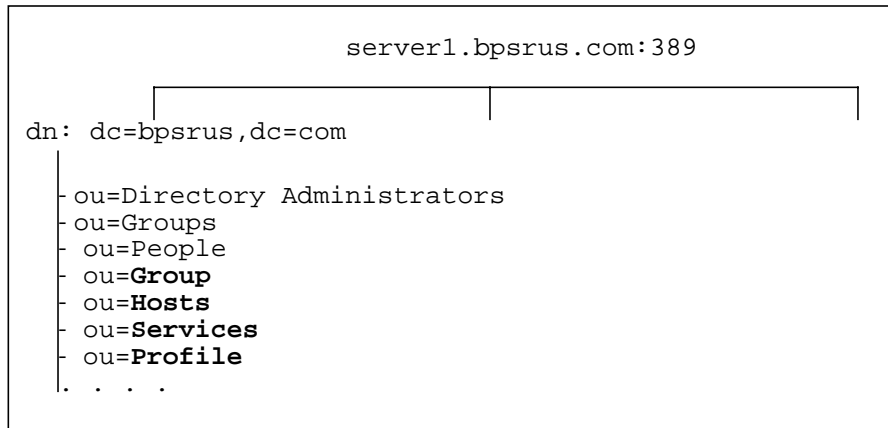
The three basic options for storing NIS data are:

- Use the directory main suffix.
- Create and use a subtree under the main suffix.
- Create and use a new directory suffix.

The following sections explain these options in greater detail.

## Example of NIS Data Stored Under the Main Suffix

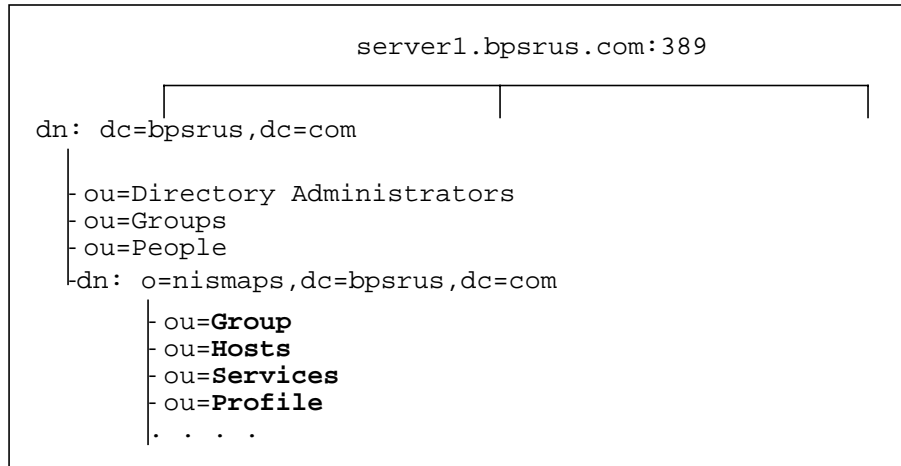
In this example, the `ou=` (organization unit) containers required to store NIS map data are created under the suffix that was created when the directory server was installed. The tree looks like this:



As shown in the above diagram, the NIS-specific containers (shown in bold type) are mixed in with the default ones created during the installation. While this topology does work, it becomes harder to determine which containers are for NIS and which ones are not. This topology also makes setting access rights less flexible, because setting them at the top affects everything underneath it.

## Example of NIS Data Stored in a Subtree

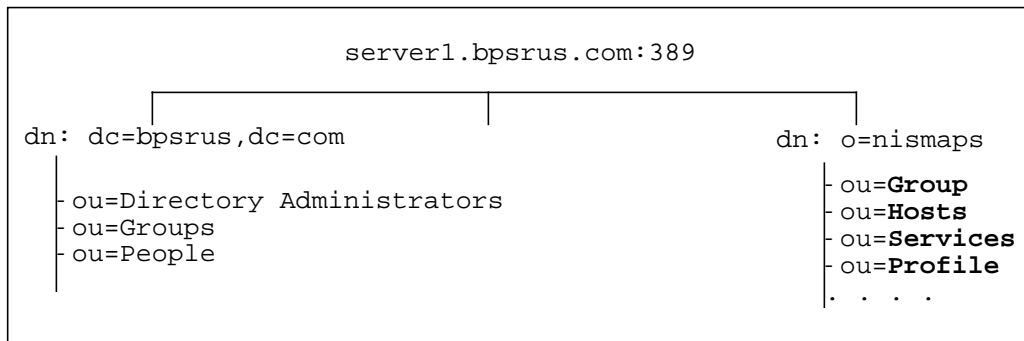
In this topology, a branch point is created below the main suffix. All NIS map data is placed under this subtree as shown in the following diagram.



The advantage of this method is that NIS map data is separated from the rest of the directory data so top level permission can be set. Also the subtree DN maintains the dc=bpsrus,dc=com part of the name for easy identification.

## Example of NIS Data Stored Under a Separate Suffix

The third method is to create a separate DIT for storing NIS data. The following diagram shows what this would look like.



The advantage of this topology is that the NIS data is isolated from other data and you can set access rights for the entire DIT. The disadvantage is that you lose the trailing part of the name, which makes it difficult to identify the domain being serviced from an entry's DN.

## Adding `nisDomainObject`

The `nisDomain` attribute must be present in the top entry of the subtree where the NIS maps are stored. Both the LDAP client search base and the name of the NIS domain (serviced by the server) are set to this top entry. To set the `nisDomain` attribute, add the `nisDomainObject` object to the top node entry as described in the next section. The schema definition for this object class looks like this:

```
objectclass nisDomainObject
  oid 1.3.6.1.1.1.2.15
  superior top
  requires
    nisDomain
```

Since this is not a standard LDAP object class, you need to add the schema definition to your server before you can create an entry of this type. This should have been done when the server was set up to support native LDAP. To verify that the server has this object class added correctly to its schema, run the following command.

```
server1# ldapsearch -b cn=schema \ objectclass=* | grep nisDomainObject
objectclasses: ( 1.3.6.1.1.1.2.15 NAME 'nisDomainObject' DESC 'User
Defined Object
```

If you do not see this line, the object class has not been added to the schema and you still need to add the schema definition before the `nisDomainObject` object can be created.

The LDIF you use to create the `nisDomainObject` object varies depending on whether you place the NIS maps at the top of the tree created during the installation, under a separate branch point of that tree, or under a new suffix. The next sections give example LDIF that can be imported to add the `nisDomainObject` object in each of these scenarios. Even though the attribute is called `nisdomain`, the client accessing the directory is not using NIS. The name is simply carried over from legacy NIS implementations.

## Directly Below the Main Suffix

The first example LDIF assumes the containers are created directly under a suffix that was created when Netscape DS 4.x was installed.

```
dn: dc=bpsrus, dc=com
changetype: modify
add: objectclass
objectclass: nisDomainObject
-
add: nisdomain
nisdomain: bpsrus.com
```

The LDIF shown here is imported to add the `nisDomainObject` object class to the `top` entry. The `nisdomain` attribute is also set here.

You can import the following LDIF to change the `nisdomain` attribute:

```
dn: dc=bpsrus,dc=com
changetype: modify
replace: nisdomain
nisdomain: east.bpsrus.com
```

---

**Note** – You do not have to match the value for `nisdomain` to the DNS domain name, but it is a common practice to do so.

---

## At the Top of a Subtree

The next example LDIF assumes you have a main suffix of `dc=bpsrus,dc=com` and you are creating a branch under it called `o=nismaps`.

```
dn: o=nismaps,dc=bpsrus,dc=com
changetype: add
objectclass: top
objectclass: organization
objectclass: nisDomainObject
o: nismaps
nisdomain: bpsrus.com
```

The following LDIF is imported to change the `nisdomain` attribute in the entry previously created:

```
dn: o=nismaps,dc=bpsrus,dc=com
changetype: modify
replace: nisdomain
nisdomain: east.bpsrus.com
```

## To a Separate Suffix

The last example LDIF assumes you are creating a new suffix for your directory server where NIS map data is stored:

```
dn: o=nismaps
changetype: add
objectclass: top
objectclass: organization
objectclass: nisDomainObject
o: nismaps
nisdomain: bpsrus.com
```

Before this LDIF is imported, you must create a new suffix as described earlier in this article. To change the `nisdomain` attribute in this topology, the following LDIF is imported:

```
dn: o=nismaps
changetype: modify
replace: nisdomain
nisdomain: east.bpsrus.com
```

---

## How LDAP Clients Find the DIT

When the client is initialized, an IP address of one or more LDAP servers and a search base is specified. This information can be specified as a command line argument to the `ldapclient` command, or in a profile generated by the `ldap_gen_profile` command. The preferred method is to generate a profile with the `ldap_gen_profile` command. The search base that is set in the profile is determined by how the tree is set up.

The `nisdomain` value that the client looks for is the name listed in the `/etc/defaultdomain` file, or one supplied with the `-d` argument to the `ldapclient` command.

The steps that the `ldapclient` command perform are:

1. Search the LDAP server's DSE to obtain the naming contexts supported in the specified directory server.
2. Search the found naming contexts for an entry containing a `nisDomainObject` object.
3. Check the found `nisdomain` attribute of the entry to verify if its value equals the value stored in the client's `/etc/defaultdomain` file.
4. Search the `ou=profile` container directly below the entry for an entry that matches the profile name provided on the command line.
5. Use the information retrieved from the profile entry to create the `/var/ldap/ldap_client_file` and `/var/ldap/ldap_client_cred` files on the client.

A key point here is that the search for the profile entry will start directly below the entry containing the `nisDomainObject` with the matching `nisdomain` value. Another important point is that you only want to have one entry with the same `nisdomain` value in the directory server. The search will stop at the first match and fail if it cannot find the specified profile which is expected to be directly below the entry with `nisDomainObject`.

## Specifying Alternate Search Paths

Client searches on the naming service database default to `ou=people`, `ou=group`, etc. based on the `SolarisSearchBaseDN` variable set in the LDAP client profile. However, different search bases can be specified for different databases. You can specify these by overriding the defaults in the profile.

To override a default, use the `-B` option of the `ldap_gen_profile` command. For example:

```
# ldap_gen_profile -P altpasswd -b o=nismaps,dc=bpsrus \  
-B "passwd:(ou=people,dc=bpsrus,dc=com)" -D cn=proxyagent,\  
ou=profile,dc=bpsrus,dc=com -w mysecret 128.100.100.1
```

In this example, the `passwd` database is accessed from an alternative path. If user account information is shared with applications other than Solaris OE clients, you should separate the `People` container from the rest of the naming service databases.

---

## Conclusion

This article described the basic structure and components which make up the LDAP DIT and provided examples that demonstrate how you would represent NIS data in this topology. A previous article titled “*Using dsimport to Convert NIS Maps to LDAP Directory Entries*”, describes the mechanics for populating the DIT once you have defined the topology you want to implement. Together, these two articles provide a foundation for your NIS to LDAP migration planning.

---

## Acknowledgements

I would like to thank Michael Haines of Sun Professional Services<sup>SM</sup> for lending his LDAP expertise to help create this article and verify its accuracy.

---

### *Author's Biography: 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 20 years of experience with the UNIX operating system and has been a Sun Engineer since 1984.*