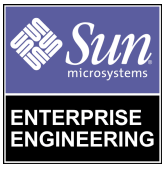
Exploring the iPlanet™ Directory Server NIS Extensions

By Tom Bialaski - Enterprise Engineering

Sun BluePrints™ OnLine - August 2000



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.: 806-6200-10
Revision 01, August 2000

Exploring the iPlanet™ Directory Server NIS Extensions

In a previous article, we examined PADL Software's *ypldapd*, which provides a gateway from Network Information Service (NIS) to Lightweight Directory Access Protocol (LDAP). This article takes a look at the *Solaris Extensions for Netscape Directory Server 4.11* which provides similar functionality to *ypldapd*, but is implemented quite differently. Instead of providing a gateway which accepts NIS calls from clients, then converts them to LDAP, the NIS Extensions provide a synchronization service between NIS maps and an LDAP directory. Understanding how the NIS extensions work is key to deciding whether to deploy them as part of your NIS to LDAP transition plans.

What Are the Extensions?

The Extensions are an add-on software package to the iPlanet™ Directory Server which provides a service that allows NIS map data to be stored in an LDAP directory, and at the same time, makes the information available to NIS clients. The service is referred to as a synchronization service since data is maintained as both NIS maps and as LDAP entries, then synchronized whenever changes are made. The architecture of the extensions is such that an entire NIS server deployment can be replaced or simply deployed as a NIS slave server to complement an existing NIS infrastructure.

When the NIS Extension software package is installed, a plug-in is added to the iPlanet Directory Server. The plug-in communicates with a Solaris Operating Environment process called `dsservd` which emulates a NIS server. NIS clients communicate with `dsservd` in the same manner they would with the native Solaris Operating Environment process `ypserv`. The NIS server emulator maintains a set of NIS maps just like a native NIS server would.

Besides being able to respond to NIS client requests, `dsservd` is able to update its NIS maps when data changes in the LDAP directory and conversely update the LDAP data when NIS maps are changed. This synchronization occurs through an inter-process communication channel between the iPlanet Directory Server plug-in and `dsservd`.

Client / Server Interaction

To better illustrate how the NIS extensions work, FIGURE 1 shows the data flow between clients and the server they are accessing.

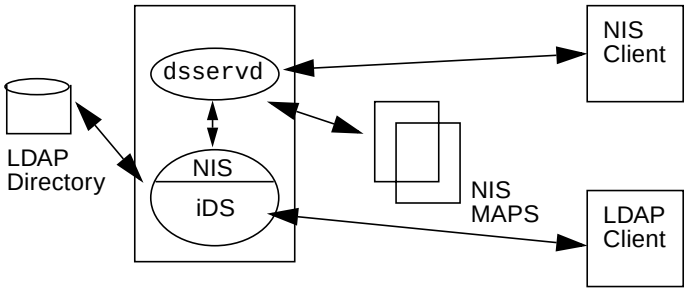


FIGURE 1 NIS Extensions Data Flow Diagram

The server shown is running the iPlanet Directory Server with NIS extensions. As you can see, the server maintains both an LDAP directory and a set of NIS maps. The `dsservd` process shown looks like a `ypserv` process to the NIS client, which is bound to it. The binding occurs either by the Broadcast method or by specifying a list of NIS servers at boot time (Specified Server method). NIS requests are serviced by consulting data in the NIS maps.

LDAP clients communicate directly with the directory server. Since the data is synchronized between the two data stores, each client sees the same view. It should be noted that the directory schema required to support the NIS extensions is some what different than the schema required to support Solaris 8 Operating Environment LDAP clients. These differences and the ramifications will be explained in a future article.

One interaction not shown in the diagram is how users update their passwords. A daemon process called `dsyppasswdd` runs on the NIS master server. It functions like the `rpc.yppasswdd` daemon, but updates the user's entry in the LDAP directory first, then synchronizes the change with the NIS maps.

NIS and LDAP Data Synchronization

Updates can be performed either on the LDAP directory or the NIS maps. When changes occur on the LDAP directory, for example, as the result of an `ldapmodify` command, the change is detected by the NIS plug-in, then passed on to the NIS maps. If the change occurs by updating an NIS map, perhaps by `makedbm`, the corresponding LDAP entries are updated. The way this is accomplished is by modifying the `Makefile` used to generate your NIS maps. A special directive is added to initiate `ldapmodify` commands which update the LDAP directory.

Since text files are generally used as the source which is used to generate NIS maps, there is a mechanism included which updates these text files to reflect changes made to the NIS maps when performed through the LDAP side.

Data Replication

Since both LDAP and NIS have their own data replication scheme, either or both methods can be deployed. LDAP has the advantage of being able to perform incremental updates, but either method will work. FIGURE 2 illustrates some possible replication scenarios.

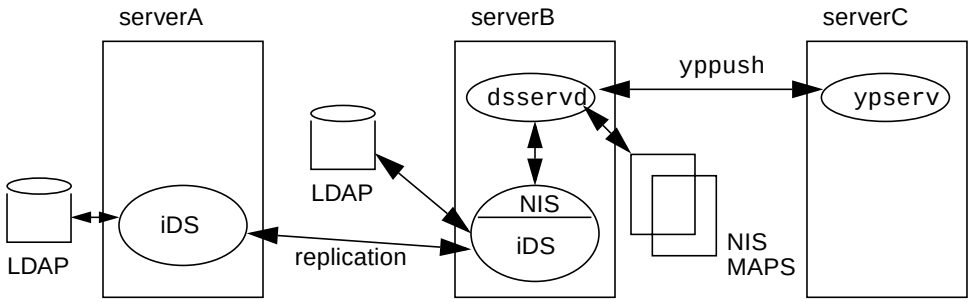


FIGURE 2 Data Replication

In this diagram, `serverA` is running the directory server with the extensions and `serverC` is running as a native NIS server. Two forms of data replication are being used. One is NIS-based and the other is LDAP-based. Also, `serverB` can run in either a NIS master or NIS slave mode. Running in slave mode has the advantage since you can simply add a slave to an existing NIS environment. However, if you run in slave mode, the LDAP directory becomes read-only.

In the master mode, the NIS maps on `serverB` are updated, then pushed to `serverC`. Changes to NIS data can occur either through standard NIS methods, such as, regenerating a NIS map with `makedbm`, or by updating the LDAP directory

using LDAP methods such as `ldapmodify` or by importing an LDIF file. Synchronization occurs when either the NIS plug-in or `dsservd` process detects a change. The changes are then propagated from one data store to the other.

NIS Extensions Initialization

The NIS extensions are contained in a `tar` file which extracts to Solaris Operating Environment packages. The `SUNWdsnis` package where the NIS extensions reside, is installed after the iPlanet Directory Server 4.11 software is installed. Both the server and extension package (*Solaris Extensions for Netscape Directory Server 4.11*) can be downloaded from the iPlanet web site at:
http://www.iplanet.com/downloads.

Initialization Overview

The following steps summarize what configuration changes need to be made.

1. Update the directory schema.
2. Examine the NIS Master's `Makefile` and modify.
3. Create the subtree topology where the NIS information is stored.
4. Import the NIS information.
5. Establish the NIS server role.
6. Set up the NIS replication policy.

Directory Schema Update

The additional object classes and attributes required to support the NIS extensions are added to the user-defined attribute and object class configuration files `slapd.user_at.conf` and `slapd.user_oc.conf`. To view these changes, type

more <*install-dir*>/*instance*/slapd.user_at.conf or
more <*install-dir*>/*instance*/slapd.user_oc.conf at the root prompt, as shown in the following two examples:

```
blueprints# more <install-dir>/instance/slapd.user_at.conf
# User defined attributes
# These attributes can be updated via LDAP by modifying the
# cn=schema schema entry. The attributes in slapd.at.conf can not
# be updated
attribute rfc822mailMember     rfc822mailMember-oid cis
attribute nisNetIdUser         1.3.6.1.4.1.42.2.27.1.1.12 ces
attribute nisNetIdGroup        1.3.6.1.4.1.42.2.27.1.1.13 ces
attribute nisNetIdHost         1.3.6.1.4.1.42.2.27.1.1.14 ces
attribute sunNisMapFullName    1.3.6.1.4.1.42.2.27.1.1.1 ces
attribute sunNisDomain 1.3.6.1.4.1.42.2.27.1.1.2 ces
. . .
blueprints#
```

```
blueprints# more <install-dir>/instance/slapd.user_oc.conf
# user defined objectclasses
# These ObjectClasses are read/writable over LDAP
# The ObjectClasses in slapd.oc.conf are read only and may not be
# updated
objectclass nismailalias
        oid 1.3.6.1.4.1.42.2.27.1.2.5
        superior top
        requires
                cn
        allows
                rfc822mailMember

objectclass nisnetid
        oid 1.3.6.1.4.1.42.2.27.1.2.6
        superior top
        requires
                cn
        allows
                nisNetIdUser,
                nisNetIdGroup,
                nisNetIdHost
. . .
blueprints#
```

`Makefile` Examination and Modification

The creation of NIS maps is determined by targets defined in `Makefile`, which by default resides in `/var/yp` on the NIS master server. The NIS extension software consults this file to determine which NIS maps are currently being used and then modifies it so a special `make` command is invoked instead of the standard `makedbm`.

The following lines in `Makefile` are modified.

```
YPDBDIR=/var/yp
MAKEDBM=$(SBINDIR)/makedbm
MKALIAS=$(YPDIR)/mkalias
```

These are changed by the software to:

```
YPDBDIR=/var/yp/ldapsynch
MAKEDBM=/opt/SUNWconn/ldap/lib/dsmakedbm
MKALIAS=/opt/SUNWconn/ldap/lib/dsmakealias
```

Based on the targets listed in `Makefile`, LDAP containers are created. For example, an organizational unit (`ou`) container is created for each target map listed below:

```
all: passwd group hosts ipnodes ethers networks rpc
services protocols \
        netgroup bootparams publickey \
        auto.master auto.home
```

Creating the Subtree

The initialization script automatically creates subtree components in the directory by issuing `ldapmodify` commands. The portion of the directory tree where these components are created is determined by the `NAMING_CONTEXT` variable. The variable can be set by un-commenting it in the `nis.mapping` file as shown below. If it is not set, the NIS domain name is used instead.

```
# The name of the NIS domain
    DOMAIN_NAME=iplanet.sun.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
#
```

Importing NIS Maps

Once the system is initialized to be a NIS server, the data contained in the NIS maps needs to be imported into the LDAP directory. This is performed by reading the text files used to generate the NIS maps, and then issuing `ldapmodify` commands to update the directory. The `dsimport` command, provided with the extensions, does this for you.

Determining the Server Role

The role of the NIS server, master or slave, is determined when the NIS extension installation script is run. The role can be changed later by running the `dsypinit -m` or `dsypinit -s` command. Like NIS, a server running the extensions can be a master of some maps and slave of others, although this is not advisable. The ownership of NIS maps can be set by modifying the `Makefile`.

LDAP Replication

NIS data stored in an LDAP directory is replicated in the normal LDAP fashion. Added security can be obtained by performing replication over a SSL channel, but this is probably not necessary if you are comfortable with the native NIS model. Like NIS, passwords are stored in *crypt* format, so a clear text version is never sent over the wire.

Conclusion

Two approaches can be taken to deploy LDAP as a naming service on existing Solaris Operating Environment NIS clients. One approach is to convert all your clients to the Solaris 8 Operating Environment native LDAP client and the other is to implement a phased deployment. The iPlanet NIS Extensions (*Solaris Extensions for Netscape Directory Server 4.11*) are a very useful tool if you choose a phased approach. However, if you are planning a Solaris 8 Operating Environment migration, you may consider switching naming services at the same time.

In the next article in this series, the native Solaris 8 Operating Environment LDAP client implementation is examined along with basics on how to configure a directory server to support them. Subsequent articles will focus data conversion and security implications.

Author's Bio: Tom Bialaski

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