



Using the LDAP to NIS+ Gateway

Tom Bialaski, Enterprise Engineering

and Michael Haines, Enterprise Services

Sun BluePrints™ OnLine—September 2003



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

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

Part No. 817-3594-10
Revision 1.0, 8/4/03
Edition: September 2003

Copyright 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology 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 document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this 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 in other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, AnswerBook2, docs.sun.com, Sun ONE, Java, Sun Open Net Environment (Sun ONE), iPlanet, SunDocs, Sun BluePrints, iPlanet, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and in other countries.

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. in the U.S. and in other countries. Products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc. Netscape is a trademark or registered trademark of Netscape Communications Corporation in the United States 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.

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 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054, Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie 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 déposée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company, Ltd.

Sun, Sun Microsystems, le logo Sun, AnswerBook2, docs.sun.com, Sun ONE, Java, Sun Open Net Environment (Sun ONE), iPlanet, SunDocs, Sun BluePrints, iPlanet, et Solaris sont des marques de fabrique ou des marques déposées de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

Toutes les marques SPARC sont utilisées sous licence et sont des marques de fabrique ou des marques déposées de SPARC International, Inc. aux Etats-Unis et dans d'autres pays. Les produits portant les marques SPARC sont basés sur une architecture développée par Sun Microsystems, Inc. Netscape est une marque de Netscape Communications Corporation aux Etats-Unis et dans d'autres pays.

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.



Please
Recycle



Adobe PostScript

Using the LDAP to NIS+ Gateway

This article contains information derived from a Sun BluePrints book titled, *LDAP in the Solaris Operating Environment – Deploying Secure Directory Services*, by Michael Haines and Tom Bialaski. This book is scheduled for publication in the Fall of 2003.

There are two approaches that you can take when transitioning from NIS+ to LDAP-based services. One approach is to replace your naming service clients with the Secured LDAP Client. The second approach is to keep your current NIS+ clients, and deploy a transition tool to gain access to LDAP naming service data.

The first approach is covered in the upcoming book in Chapter 4 “Deploying Solaris OE LDAP Naming Services”. The second approach, using the NIS+ to LDAP Gateway, is discussed in the following sections of this article:

- “What is a Gateway?” on page 2
- “Using the Gateway as a Transition Tool” on page 4
- “Configuring the Sun ONE Directory Server Software as a Configuration Server for `rpc.nisd`” on page 20
- “How NIS+ Data is Mapped to LDAP” on page 22
- “NIS+ to LDAP Migration Example” on page 33
- “Testing and Troubleshooting the NIS+ Gateway” on page 37

This article is intended for IT architects and administrators who have some experience with NIS+, and who are interested in deploying LDAP using the Sun ONE Directory Server 5.2 software.

What is a Gateway?

A gateway is a service that provides a mapping of one type of data to another. In respect to naming services, a gateway is used to map NIS+ data to LDAP data. The NIS+ gateway maintains a copy of NIS+ data, most of which is stored in a LDAP directory. The copy of the data is updated whenever the corresponding LDAP entries are updated. It can also be updated using the native naming service tools.

To function transparently to clients, a modified version of the native server `rpc.nisd`, is used to service the clients. The modified server references a mapping file to determine how to store the native naming service data in LDAP and how to retrieve it so it can be presented to the client. The modified server is also responsible for maintaining cache consistency.

The advantage of the new gateway over the previous NIS Extensions implementation is that the service is decoupled from any particular LDAP server. The NIS Extensions were implemented partly as a directory plug-in that only worked with a specific directory server version (Netscape Directory Server 4.1x). The process that provided the native naming service (`dsypservd`) had to run on the same system as the directory server.

The new gateway technology uses a Solaris OE process that is a standalone process from the directory server process. This allows the gateway to run on a different server than the directory server, although you can run them on the same server if you want to. The only dependency is the operating environment. FIGURE 0-1 shows what a gateway deployment might look like.

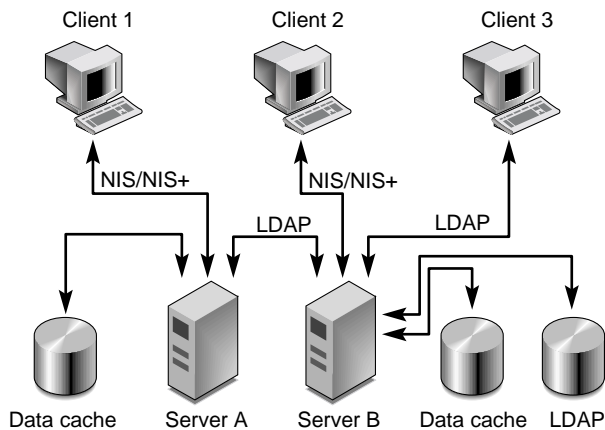


FIGURE 0-1 Gateway Deployment

In FIGURE 0-1, Client 1 is a native NIS+ client that communicates to Server A which is running the gateway software. In this case, only cached data is maintained on the server. When data updates occur, the cache is refreshed from the directory server that resides on Server B. The native NIS+ client called Client 2 communicates directly to a server where the directory server resides. The third client, Client 3, is running native LDAP and is accessing the same information as the other clients.

NIS+ Gateway Components

FIGURE 0-1 shows how the gateway is implemented. The software for this service is contained in the following packages:

- SUNWnistr
- SUNWnisu

These packages also contain the software for native NIS and NIS+. They are loaded as part of the core installation, so should be available. However, to run the NIS+ Gateway software, a directory server must exist somewhere on the network. In theory, any LDAP v3-compliant directory server should work, but only the bundled Sun ONE directory Server software is discussed here. This is because several additional configuration steps are required to configure other directory servers, and those steps are beyond the scope of this article.

There are only three visible components of the NIS+ Gateway software:

- `/usr/sbin/rpc.nisd`
- `/etc/default/rpc.nisd`
- `/var/nis/NIS+LDAPmapping`

The same `rpc.nisd` process is used whether you are running the gateway or running native NIS+ without the gateway. The `rpc.nisd` process determines the configuration by the presence of the `/var/nis/NIS+LDAPmapping` file. If a file with that exact name does not exist, the `rpc.nisd` process will not use the directory server as a back-end store.

The `/etc/default/rpc.nisd` file contains information about what directory server to use and where to search for information it contains. There are many parameters that can be configured in this file, as described in a following section.

The `NIS+LDAPmapping` file describes how the NIS+ data is mapped to LDAP attributes and object classes. The mapping file is quite complex, but in most deployments, only a slight modification is required. For reference, all the relevant parameters are described.

Using the Gateway as a Transition Tool

The following tasks are recommended:

1. The Solaris 9 OE `rpc.nisd(1M)` can be configured to use an LDAP server as its data repository. In this way, the normal NIS+ database becomes a cache for the LDAP data, with configurable time to live (TTL) values. The cache improves performance, and enables the `rpc.nisd` to serve NIS+ data even if communication with the LDAP server is interrupted temporarily.
2. Two configuration files, `rpc.nisd(4)` and `NIS+LDAPmapping(4)`, control communication between `rpc.nisd` and the LDAP server, and the way that NIS+ table entries are mapped to LDAP entries. A template configuration file, `/var/nis/NIS+LDAPmapping.template`, covers all standard NIS+ tables, and can be used as a basis when creating a customized NIS+ to LDAP mapping.
3. A test utility, `nislmaptest(1M)`, can be used to try out mapping configurations without affecting NIS+ data.
4. The NIS+ `passwd.org_dir` table stores passwords in the UNIX crypt format, so the LDAP server must be set up to use crypt format for the `userPassword` attribute for those accounts that are shared with NIS+.
5. In general, migration from NIS+ to LDAP starts by installing and configuring one or more LDAP servers to support the Solaris OE name service clients.
6. Install the Solaris 9 OE on the NIS+ master, and configure `rpc.nisd` on the NIS+ master to map NIS+ data to and from LDAP. The `rpc.nisd` has options that enable uploading all NIS+ data to LDAP.
7. Only the NIS+ master needs to run the Solaris 9 OE. All other NIS+ servers and clients can remain on earlier Solaris OE releases (or whatever OS they are currently using).
8. Once a complete LDAP name service environment exists, conversion of NIS+ clients to LDAP clients can start. While this conversion is going on, NIS+ and LDAP name service clients share the same data, and the NIS+ master `rpc.nisd` keeps the data in sync.

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

The following sections examine the configuration files and the parameters they contain.

rpc.nisd - Configuration File for NIS+ Service Daemon

The `/etc/default/rpc.nisd` file specifies configuration information for the `rpc.nisd` server. Configuration information can come from a combination of three places (listed in order of precedence):

1. From the `rpc.nisd` command line
2. From the `/etc/default/rpc.nisd` file
3. From information stored in an LDAP directory

Although many configuration parameters can be set, in most cases the defaults will work fine.

Most of the parameters used in the `/etc/default/rpc.nisd` file have the same names as the corresponding LDAP attributes. Some of the initialization parameters do not have equivalents because they are used to locate and connect to the LDAP server to retrieve the configuration data.

The parameters can be placed in three categories:

- Initialization parameters
- Parameters used to retrieve data
- Parameters to initiate some type of action

Each of these types is described in the following sections.

Initialization Parameters

The following attributes are not part of the `nisplusLDAPconfig` object class and can only be set locally, either in `rpc.nisd(4)`, or from the command line. These are used only for loading configuration data from the LDAP server. If all the configuration data is set in `NIS+LDAPmapping(4)` and `rpc.nisd(4)`, these parameters need not be set.

Note – If the LDAP service is unavailable, `rpc.nisd(1M)` will wait until it can successfully read the configuration data before offering the NIS+ service.

- `nisplusLDAPconfigDN` – The DN for configuration information. If empty, all other `nisplusLDAPConfig*` values are ignored with the expectation that all attributes are specified in this file or on the command line. When `nisplusLDAPconfigDN` is not specified at all, the DN is derived from the NIS+ domain name by default. If the domain name is `x.y.z.`, the default `nisplusLDAPconfigDN` is:

`nisplusLDAPconfigDN=dc=x,dc=y,dc=z`

- `nisplusLDAPconfigPreferredServerList` - The list of servers to use for the configuration phase. There is no default. The following is an example of a value for `nisplusLDAPconfigPreferredServerList`:

`nisplusLDAPconfigPreferredServerList=127.0.0.1:389`

- `nisplusLDAPconfigAuthenticationMethod` - The authentication method used to obtain the configuration information. The recognized values for `nisplusLDAPconfigAuthenticationMethod` are:
 - `none` - No authentication attempted.
 - `simple` - Password of proxy user sent in the clear to the LDAP server.
 - `sasl/cram-md5` - Use SASL/CRAM-MD5 authentication. This authentication method may not be supported by all LDAP servers. A password must be supplied.
 - `sasl/digest-md5` - Use SASL/DIGEST-MD5 authentication. This authentication method may not be supported by all LDAP servers. A password must be supplied.

There is no default value. The following is an example of a value for `nisplusLDAPconfigAuthenticationMethod`:

`nisplusLDAPconfigAuthenticationMethod=simple`

- `nisplusLDAPconfigTLS` - The transport layer security used for the connection to the server. The recognized values are:
 - `none` - No encryption of transport layer data. This is the default value.
 - `ssl` - SSL encryption of transport layer data. The directory server must be configured for SSL and a certificate database is required.

Export and import control restrictions may limit the availability of transport layer security.

- `nisplusLDAPconfigTLSCertificateDBPath` - The name of the file containing the certificate database. The default path is `/var/nis`, and the default file name is `cert7.db`. The certificate database is set up the same way as with the Secured LDAP Client.
- `nisplusLDAPconfigProxyUser` - The proxy user used to obtain configuration information. There is no default value. If the value ends with a comma, the value of the `nisplusLDAPconfigDN` attribute is appended. For example:
`nisplusLDAPconfigProxyUser=cn=nisplusAdmin,ou=People,`
- `nisplusLDAPconfigProxyPassword` - The password that should be supplied to LDAP for the proxy user when the authentication method requires one. In order to avoid having this password publicly visible on the machine, the password should only appear in the configuration file, and the file should have an appropriate owner, group, and file mode. There is no default value.

Data Retrieval Parameters

- `preferredServerList` - The list of servers to use when reading or writing mapped NIS+ data from or to LDAP. There is no default value. Example:
`preferredServerList=127.0.0.1:389`
- `authenticationMethod` - The authentication method to use when reading or writing mapped NIS+ data from or to LDAP. For recognized values, see the `LDAPconfigAuthenticationMethod` attribute. There is no default value. Example:
`authenticationMethod=simple`
- `nisplusLDAPTLS` - The transport layer security to use when reading or writing NIS+ data from or to LDAP. For recognized values, see the `nisplusLDAPconfigTLS` attribute. The default value is none. Note that export and import control restrictions may limit the strength of encryption available to the transport layer security.
- `nisplusLDAPTLSCertificateDBPath` - The name of the file containing the certificate DB. For recognized and default values see the `nisplusLDAPconfigTLSCertificateDBPath` attribute.
- `defaultSearchBase` - The default portion of the DN to use when reading or writing mapped NIS+ data from or to LDAP. The default is derived from the value of the `baseDomain` attribute, which in turn usually defaults to the NIS+ domain name. If `nisplusLDAPbaseDomain` has the value `x.y.z`, the default `defaultSearchBase` is `dc=x,dc=y,dc=z`. Sample attribute value:
`defaultSearchBase=dc=somewhere,dc=else`
- `nisplusLDAPbaseDomain` - The domain to append when NIS+ object names are not fully qualified. The default is the domain the `rpc.nisd` daemon is serving, or the first such domain, if there is more than one candidate.
- `nisplusLDAPproxyUser` - Proxy user used by the `rpc.nisd` to read or write from or to LDAP. There is no default value. If the value ends in a comma, the value of the `defaultSearchBase` attribute is appended. Example:
`nisplusLDAPproxyUser=cn=nisplusAdmin,ou=People,`
- `nisplusLDAPproxyPassword` - The password that should be supplied to LDAP for the proxy user when the authentication method so requires. In order to avoid having this password publicly visible on the machine, the password should only appear in the configuration file, and the file should have an appropriate owner, group, and file mode. There is no default value.
- `nisplusLDAPbindTimeout`
- `nisplusLDAPsearchTimeout`
- `nisplusLDAPmodifyTimeout`
- `nisplusLDAPaddTimeout`
- `nisplusLDAPdeleteTimeout`

The five attributes listed above, establish timeouts for LDAP bind, search, modify, add, and delete operations, respectively. The default value is 15 seconds for each one. Floating point values are allowed.

- `nisplusLDAPsearchTimeLimit` - Establish a value for the `LDAP_OPT_TIMELIMIT` option, which suggests a time limit for the search operation on the LDAP server. The server may impose its own constraints on possible values. Refer to LDAP server documentation. The default is the `nisplusLDAPsearchTimeout` value. Only integer values are allowed.

The `nisplusLDAPsearchTimeout` limits the amount of time the client `rpc.nisd` waits for completion of a search operation, so setting the `nisplusLDAPsearchTimeLimit` larger than the `nisplusLDAPsearchTimeout` is not recommended.

- `nisplusLDAPsearchSizeLimit` - Establish a value for the `LDAP_OPT_SIZELIMIT` option, which suggests a size limit, in number of entries, for the search results on the LDAP server. The server may impose its own constraints on possible values. Refer to LDAP server documentation. The default is zero, which means unlimited. Only integer values are allowed.
- `nisplusLDAPfollowReferral` - Determines if the `rpc.nisd` should follow referrals or not. Recognized values are `yes` and `no`. The default value is `no`.
- `nisplusNumberOfServiceThreads` - Sets the maximum number of RPC service threads that the `rpc.nisd` may use. Note that the `rpc.nisd` can create additional threads for certain tasks, so that the actual number of threads running can be larger than the `nisplusNumberOfServiceThreads` value.

The value of this attribute is a decimal integer from 0 (zero) to $(2^{31})-1$, inclusive. 0, which is the default, sets the number of service threads to three plus the number of CPUs available when the `rpc.nisd` daemon starts. Example:

```
nisplusNumberOfServiceThreads=16
```

Action-Related Parameters

The following attributes specify the action to be taken when some event occurs. The values are all of the form `event=action`. The default action is the first one listed for each event.

Note – A complete list of parameters is provided, although only the ones with LDAP in their names are directly related to the NIS+ to LDAP gateway.

- `nisplusLDAPinitialUpdateAction` - Provides the optional capability to update all NIS+ data from LDAP, or vice versa, when the `rpc.nisd` starts. Depending on various factors such as both NIS+ and LDAP server and network performance, as well as the amount of data to be uploaded or downloaded, these operations can consume significant CPU and memory resources. During upload

and download, the `rpc.nisd` has not yet registered with `rpcbind`, and provides no NIS+ service. When data is downloaded from LDAP, any new items added to the `rpc.nisd`'s database get a TTL for an initial load. See the description for the `nisplusLDAPentryTtl` attribute on `NIS+LDAPmapping(4)`.

- `none` - No initial update in either direction. This is the default.
- `from_ldap` - Causes the `rpc.nisd` to fetch data for all NIS+ objects it serves, and for which mapping entries are available, from the LDAP repository.
- `to_ldap` - The `rpc.nisd` writes all NIS+ objects for which it is the master server, and for which mapping entries are available, to the LDAP repository.
- `nisplusLDAPinitialUpdateOnly` - Use in conjunction with `nisplusLDAPinitialUpdateAction`.
 - `no` - Following the initial update, the `rpc.nisd` starts serving NIS+ requests. This is the default.
 - `yes` - The `rpc.nisd` exits after the initial update. This value is ignored if specified together with `nisplusLDAPinitialUpdateAction=none`.
- `nisplusLDAPretrieveErrorAction` - If an error occurs while trying to retrieve an entry from LDAP, one of the following actions can be selected:
 - `use_cached` - Action according to `nisplusLDAPrefreshError` below. This is the default.
 - `retry` - Retry the retrieval the number of time specified by `nisplusLDAPretrieveErrorAttempts`, with the `nisplusLDAPretrieveErrorTimeout` value controlling the wait between each attempt.
 - `try_again`
 - `unavail`
 - `no_such_name`
 - `Return` - with `NIS_TRYAGAIN`, `NIS_UNAVAIL`, or `NIS_NOSUCHNAME`, respectively, to the client.

Note that the client code may not be prepared for this and can react in unexpected ways.
- `nisplusLDAPretrieveErrorAttempts` - The number of times a failed retrieval should be retried. The default is unlimited. The `nisplusLDAPretrieveErrorAttempts` value is ignored unless `nisplusLDAPretrieveErrorAction=retry`.
- `nisplusLDAPretrieveErrorTimeout` - The timeout (in seconds) between each new attempt to retrieve LDAP data. The default is 15 seconds. The value for `nisplusLDAPretrieveErrorTimeout` is ignored unless `nisplusLDAPretrieveErrorAction=retry`.
- `nisplusLDAPstoreErrorAction` - An error occurred while trying to store data to the LDAP repository.

- `retry` - Retry operation `nisplusLDAPstoreErrorAttempts` times with `nisplusLDAPstoreErrorTimeout` seconds between each attempt. Note that this might tie up a thread in the `rpc.nisd` daemon.
- `system_error` - Return `NIS_SYSTEMERROR` to the client.
- `unavail` - Return `NIS_UNAVAIL` to the client. Note that the client code may not be prepared for this and can react in unexpected ways.
- `nisplusLDAPstoreErrorAttempts` - The number of times a failed attempt to store should be retried. The default is unlimited. The value for `nisplusLDAPstoreErrorAttempts` is ignored unless `nisplusLDAPstoreErrorAction=retry`.
- `nisplusLDAPstoreErrortimeout` - The timeout, in seconds, between each new attempt to store LDAP data. The default is 15 seconds. The `nisplusLDAPstoreErrortimeout` value is ignored unless `nisplusLDAPstoreErrorAction=retry`.
- `nisplusLDAPrefreshErrorAction` - An error occurred while trying to refresh a cache entry.
 - `continue_using` - Continue using expired cache entry, if one is available. Otherwise, the action is `retry`. This is the default.
 - `retry` - Retry operation `nisplusLDAPrefreshErrorAttempts` times with `nisplusLDAPrefreshErrorTimeout` seconds between each attempt. Note that this may tie up a thread in the `rpc.nisd` daemon.
 - `cache_expired`
 - `tryagain`
 - Return - `NIS_CACHEEXPIRED` or `NIS_TRYAGAIN`, respectively, to the client. Note that the client code may not be prepared for this and could can react in unexpected ways.
- `nisplusLDAPrefreshErrorAttempts` - The number of times a failed refresh should be retried. The default is unlimited. This applies to the `retry` and `continue_using` actions, but for the latter, only when there is no cached entry.
- `nisplusLDAPrefreshErrorTimeout` - The timeout (in seconds) between each new attempt to refresh data. The default is 15 seconds. The value for `nisplusLDAPrefreshErrorTimeout` applies to the `retry` and `continue_using` actions.
- `nisplusThreadCreationErrorAction` - The action to take when an error occurred while trying to create a new thread. This only applies to threads controlled by the `rpc.nisd` daemon not to RPC service threads. An example of threads controlled by the `rpc.nisd` daemon are those created to serve `nis_list(3NSL)` with `callback`, as used by `niscat(1)` to enumerate tables.
 - `pass_error` - Pass on the thread creation error to the client, to the extent allowed by the available NIS+ error codes. The error might be `NIS_NOMEMORY`, or another resource shortage error. This action is the default.

- `retry` - Retry operation `nisplusThreadCreationErrorAttempts` times, waiting `nisplusThreadCreationErrorTimeout` seconds between each attempt. Note that this might tie up a thread in the `rpc.nisd` daemon.
- `nisplusThreadCreationErrorAttempts` - The number of times a failed thread creation should be retried. The default is unlimited. The value for `nisplusThreadCreationErrorAttempts` is ignored unless the `nisplusThreadCreationErrorAction=retry`.
- `nisplusThreadCreationErrorTimeout` - The number of seconds to wait between each new attempt to create a thread. The default is 15 seconds. Ignored unless `nisplusThreadCreationErrorAction=retry`.
- `nisplusDumpError` - An error occurred during a full dump of an NIS+ directory from the master to a replica. The replica can:
 - `retry` - Retry operation `nisplusDumpErrorAttempts` times waiting `nisplusDumpErrorTimeout` seconds between each attempt. Note that this may tie up a thread in the `rpc.nisd`.
 - `rollback` - Try to roll back the changes made so far before retrying per the `retry` action. If the rollback fails or cannot be performed due to the selected `resyncServiceAction` level, the `retry` action is selected.
- `nisplusDumpErrorAttempts` - The number of times a failed full dump should be retried. The default is unlimited. When the number of retry attempts has been used up, the full dump is abandoned, and will not be retried again until a resync fails because no update time is available.
- `nisplusDumpErrorTimeout` - The number of seconds to wait between each attempt to execute a full dump. The default is 120 seconds.
- `nisplusResyncService` - Type of NIS+ service to be provided by a replica during resync, that is, data transfer from NIS+ master to NIS+ replica. This includes both partial and full resyncs.
 - `from_copy` - Service is provided from a copy of the directory to be resynced while the resync is in progress. Rollback is possible if an error occurs. Note that making a copy of the directory might require a significant amount of time, depending on the size of the tables in the directory and available memory on the system.
 - `directory_locked` - While the resync for a directory is in progress, it is locked against access. Operations to the directory are blocked until the resync is done. Rollback is not possible.
 - `from_live` - The replica database is updated in place. Rollback is not possible. If there are dependencies between individual updates in the resync, clients might be exposed to data inconsistencies during the resync. In particular, directories or tables might disappear for a time during a full dump.
- `nisplusUpdateBatching` - How updates should be batched together on the master.

- `accumulate` – Accumulate updates for at least `nisplusUpdateBatchingTimeout` seconds. Any update that comes in before the timeout has occurred will reset the timeout counter. Thus, a steady stream of updates less than `nisplusUpdateBatchingTimeout` seconds apart could delay pinging replicas indefinitely.
- `bounded_accumulate` – Accumulate updates for at least `nisplusUpdateBatchingTimeout` seconds. The default value for timeout is 120 seconds. Incoming updates do not reset the timeout counter, so replicas will be informed once the initial timeout has expired.
- `none` – Updates are not batched. Instead, replicas are informed immediately of any update. While this should maximize data consistency between master and replicas, it can also cause considerable overhead on both master and replicas.
- `nisplusUpdateBatchingTimeout` – The minimum time (in seconds) during which to accumulate updates. Replicas will not be pinged during this time. The default is 120 seconds.
- `nisplusLDAPmatchFetchAction` – An NIS+ match operation, that is, any search other than a table enumeration, will encounter one of the following situations:
 - Table believed to be entirely in cache, and all cached entries are known to be valid. The cached tabled data is authoritative for the match operation.
 - Table wholly or partially cached, but there may be individual entries that have timed out.
 - No cached entries for the table. Always attempt to retrieve matching data from LDAP. When the table is wholly or partially cached, the action for the `nisplusLDAPmatchFetchAction` attribute controls whether or not the LDAP repository is searched:
 - `no_match_only` – Only go to LDAP when there is no match at all on the search of the available NIS+ data, or the match includes at least one entry that has timed out.
 - `always` – Always make an LDAP lookup.
 - `never` – Never make an LDAP lookup.
- `nisplusMaxRPCRecordSize` – Sets the maximum RPC record size that NIS+ can use over connection-oriented transports. The minimum record size is 9000, which is the default. The default value will be used in place of any value less than 9000. The value of this attribute is a decimal integer from 9000 to 2^{31} , inclusive.

Storing Configuration Attributes in LDAP

Most attributes previously described, as well as those from the `NIS+LDAPmapping(4)` man page, can be stored in LDAP. In order to do so, you must add definitions to your LDAP server. The definitions, provided here, are described in LDIF format, suitable for use with the `ldapadd(1)` command. The attribute and object class OIDs are examples only.

Using LDAP to Store Configuration Data

Except for the information required to locate and search an LDAP configuration server, the NIS+ Gateway configuration parameters can be stored in an LDAP directory. Most of the information is stored in the `nisplusLDAPconfig` object class, which is shown below.

The nisplusLDAPconfig Object Class:

```
NAME 'nisplusLDAPconfig'  
DESC 'NIS+/LDAP mapping configuration'  
MUST  
cn  
MAY  
preferredServerList  
defaultSearchBase  
authenticationMethod  
nisplusLDAPTLS  
nisplusLDAPTLSCertificateDBPath  
nisplusLDAPproxyUser  
nisplusLDAPproxyPassword  
nisplusLDAPinitialUpdateAction  
nisplusLDAPinitialUpdateOnly  
nisplusLDAPretrieveErrorAction  
nisplusLDAPretrieveErrorAttempts  
nisplusLDAPretrieveErrorTimeout  
nisplusLDAPstoreErrorAction  
nisplusLDAPstoreErrorAttempts  
nisplusLDAPstoreErrorTimeout  
nisplusLDAPrefreshErrorAction  
nisplusLDAPrefreshErrorAttempts  
nisplusLDAPrefreshErrorTimeout  
nisplusNumberOfServiceThreads  
nisplusThreadCreationErrorAction  
nisplusThreadCreationErrorAttempts  
nisplusThreadCreationErrorTimeout  
nisplusDumpErrorAction  
nisplusDumpErrorAttempts  
nisplusDumpErrorTimeout  
nisplusResyncService  
nisplusUpdateBatching  
nisplusUpdateBatchingTimeout  
nisplusLDAPmatchFetchAction  
nisplusLDAPbaseDomain  
nisplusLDAPdatabaseIdMapping  
nisplusLDAPentryTtl  
nisplusLDAPobjectDN  
nisplusLDAPcolumnFromAttribute  
nisplusLDAPattributeFromColumn
```

Most of the attributes that the `nisplusLDAPconfig` object class can contain are not included in any Sun ONE Directory Server schema files. The following attributes are also used by the `DUAconfigProfile` object class and are included when you run the `idsconfig` script to set up the directory server to support native LDAP clients.

- preferredServerList
- defaultSearchBase
- authenticationMethod

The last six attributes shown in the previous example do not equate to configuration parameters in the `/etc/default/rpc.nisd` file. These are used to map data from NIS+ to LDAP.

The following are the schema definitions for the attributes used in the `nisplusLDAPconfig` object class. These are included for reference only. Instructions for using LDAP to store configuration parameters are provided in the upcoming Sun BluePrints book.

Attributes shared by `DUAconfigProfile`:

```
NAME 'defaultSearchBase'
DESC 'Default LDAP base DN used by a DUA'
EQUALITY distinguishedNameMatch
SYNTAX SINGLE-VALUE )

NAME 'preferredServerList'
DESC 'Preferred LDAP server host addresses used by DUA'
EQUALITY caseIgnoreMatch \
SYNTAX SINGLE-VALUE

NAME 'authenticationMethod' \
DESC 'Authentication method used to contact the DSA'
EQUALITY caseIgnoreMatch \
SYNTAX SINGLE-VALUE )
```

Attributes Requiring Configuration:

```
NAME nisplusLDAPTLS
DESC Transport Layer Security
SYNTAX SINGLE-VALUE

NAME nisplusLDAPTLSCertificateDBPath
DESC Certificate file
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPproxyUser'
DESC 'Proxy user for data store/retrieval'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPproxyPassword'
DESC 'Password/key/shared secret for proxy user'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPbaseDomain'
DESC 'Default domain name used in NIS+/LDAP mapping'
SYNTAX SINGLE-VALUE
```

Note – These attributes can be set locally in `/etc/default/rpc.nisd`.

Attributes Generally Not Requiring Configuration:

```
NAME 'nisplusNumberOfServiceThreads'
DESC 'Max number of RPC service threads'
SYNTAX SINGLE-VALUE

NAME 'nisplusThreadCreationErrorAction' \
DESC 'Action when a non-RPC-service thread creation fails' \
SYNTAX SINGLE-VALUE

NAME 'nisplusThreadCreationErrorAttempts'
DESC 'Number of times to retry thread creation'
SYNTAX SINGLE-VALUE

NAME 'nisplusThreadCreationErrorTimeout'
DESC 'Timeout between each thread creation attempt'
SYNTAX SINGLE-VALUE

NAME 'nisplusDumpErrorAction'
DESC 'Action when a NIS+ dump fails'
SYNTAX SINGLE-VALUE

NAME 'nisplusDumpErrorAttempts'
DESC 'Number of times to retry a failed dump'
SYNTAX SINGLE-VALUE

NAME 'nisplusDumpErrorTimeout'
DESC 'Timeout between each dump attempt'
SYNTAX SINGLE-VALUE

NAME 'nisplusResyncService'
DESC 'Service provided during a resync'
SYNTAX SINGLE-VALUE

NAME 'nisplusUpdateBatching'
DESC 'Method for batching updates on master'
SYNTAX SINGLE-VALUE

NAME 'nisplusUpdateBatchingTimeout'
DESC 'Minimum time to wait before pinging replicas'
SYNTAX SINGLE-VALUE )
```

Error Action Attributes:

```
NAME 'nisplusLDAPinitialUpdateAction'
DESC 'Type of initial update'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPinitialUpdateOnly'
DESC 'Exit after update ?'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPretrieveErrorAction' \
DESC 'Action following an LDAP search error'
SYNTAX SINGLE-VALUE )

NAME 'nisplusLDAPretrieveErrorAttempts'
DESC 'Number of times to retry an LDAP search'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPretrieveErrorTimeout'
DESC 'Timeout between each search attempt'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPstoreErrorAction'
DESC 'Action following an LDAP store error'
SYNTAX 26 SINGLE-VALUE

NAME 'nisplusLDAPstoreErrorAttempts'
DESC 'Number of times to retry an LDAP store'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPstoreErrorTimeout'
DESC 'Timeout between each store attempt'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPrefreshErrorAction'
DESC 'Action when refresh of NIS+ data from LDAP fails'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPrefreshErrorAttempts'
DESC 'Number of times to retry an LDAP refresh'
SYNTAX SINGLE-VALUE

NAME 'nisplusLDAPrefreshErrorTimeout'
DESC 'Timeout between each refresh attempt'
SYNTAX SINGLE-VALUE
```

Attributes Used for Mapping Data:

```
NAME 'nisplusLDAPmatchFetchAction'  
DESC 'Should pre-fetch be done ?'  
SYNTAX SINGLE-VALUE  
  
NAME 'nisplusLDAPdatabaseIdMapping'  
DESC 'Defines a database id for a NIS+ object'  
SYNTAX  
  
NAME 'nisplusLDAPentryTtl'  
DESC 'TTL for cached objects derived from LDAP'  
SYNTAX  
  
NAME 'nisplusLDAPobjectDN'  
DESC 'Location in LDAP tree where NIS+ data is stored'  
SYNTAX  
  
NAME 'nisplusLDAPcolumnFromAttribute'  
DESC 'Rules for mapping LDAP attributes to NIS+ columns'  
SYNTAX  
  
NAME 'nisplusLDAPattributeFromColumn'  
DESC 'Rules for mapping NIS+ columns to LDAP attributes'  
SYNTAX
```

Configuring the Sun ONE Directory Server Software as a Configuration Server for `rpc.nisd`

The basic tasks for configuring the Sun ONE Directory Server software are:

1. Update the directory schema.
2. Create an LDAP entry that contains the `nisplusLDAPconfig` object class and associated parameters.
3. Modify the `/etc/default/rpc.nisd` file.

Each task is described in detail.

▼ Task 1 – To Update the Schema

1. **Obtain the `99nisplusLDAPconfig.ldif` file form the downloadable file called `ldap-schemas.tar.gz`.**
See “Obtaining the Downloadable Files for This Book” on page vii.
2. **Copy the schema file to your directory server config directory.**

```
# cp 99nisplusLDAPconfig.ldif /var/mps/serverroot/instance/  
config/schema
```

3. **Stop and restart the directory server.**

```
# directoryserver stop  
# directoryserver start
```

▼ Task 2 – To Create a Configuration Entry

1. Create an LDIF file with the appropriate parameters.

```
# vi nisplusConfig.ldif
dn: cn=example.com,dc=example,dc=com
cn: domain
objectClass: top
objectClass: nisplusLDAPconfig
nisplusLDAPproxyUser: cn=proxyagent,ou=profile,dc=example,dc=com
nisplusLDAPproxyPassword: mysecret
nisplusLDAPbaseDomain: example.com
nisplusNumberOfServiceThreads: 32
.
.
.
```

2. Import the LDIF file.

```
# ldapmodify -a -D "cn=directory manager" -w mypassword -f \
nisplusConfig.ldif
```

▼ Task 3 – To Modify `rpc.nisd`

1. At a minimum, change the following parameters:

```
# vi /etc/default/rpc.nisd
nisplusLDAPconfigDN=dc=example,dc=com
nisplusLDAPconfigPreferredServerList=125.148.181.130
nisplusLDAPconfigAuthenticationMethod=simple
nisplusLDAPconfigProxyUser=cn=proxyagent,ou=profile,dc=example,d
c=com
nisplusLDAPconfigProxyPassword=mysecret
```

Note – The example shown here uses a simple LDAP bind. This is only recommended if `rpc.nisd` and the directory server are running on the same server.

How NIS+ Data is Mapped to LDAP

To be able to access the same data from both NIS+ clients and LDAP clients, the data NIS+ tables need to be represented as LDAP object classes and attributes. The object classes for the most part have been defined in RFC 2307, which is the same data structure used to map NIS data. However, because you may wish to store NIS+ data that does not have an NIS counterpart, additional object classes and attributes might need to be defined.

The mapping of NIS+ data to LDAP is not a one-to-one relation. That is, the rows and columns of NIS+ tables are not translated directly into LDAP attributes. If this were the case, a generic LDAP representation of NIS+ tables could be used for all data. Implementing mapping that way would not leverage the power of LDAP, because one of its strengths is the ability to deal with complex data.

In most cases, the default mappings work fine. Default mappings are defined in a file called `/var/nis/NIS+LDAPmapping.template`. The defaults can be overwritten by configuration data stored in LDAP, or by command-line arguments specified when `rpc.nisd(1M)` is started. To start the mapping function, `rpc.nisd` looks for the presence of a `/var/nis/NIS+LDAPmapping` file. An alternative file can be referenced by specifying the `-m` switch.

Not all NIS+ table entry mappings are defined in the RFC 2307bis specification, so additional schema definitions can be added to your directory server. These schema definitions include:

- `timezone` attributes and object classes
- `client_info` attributes and object classes
- NIS+ object data
- Principals and netnames
- Non-standard data

Note – If you choose not to deploy additional schema definitions, it does not mean they will not be available to NIS+ clients. The information will still be available in the data cache, but it will not be stored in LDAP.

timezone Schema

If you choose to maintain time zone information in LDAP, your directory server needs to be configured with an additional schema definition. After the schema definition is added, you must uncomment the lines referring to `timezone` in the `NIS+LDAPmapping` file to activate LDAP storage of the information. Below is the LDIF representation of the `nisplusTimeZone` attribute and the `nisplusTimeZoneData` object class.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.15.0
    NAME 'nisplusTimeZone'
    DESC 'tzone column from NIS+ timezone table'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.42.2.27.5.42.42.16.0
    NAME 'nisplusTimeZoneData'
    DESC 'NIS+ timezone table data'
    SUP top STRUCTURAL MUST ( cn )
    MAY ( nisplusTimeZone $ description ) )
```

Below is a sample LDIF to create the container for the time zone data.

```
dn: ou=Timezone,dc=example,dc=com
ou: Timezone
objectClass: top
objectClass: organizationalUnit
```

client_info Schema

If you wish to store the `client_info` table in LDAP, you need to update your directory server schema files. After you update the schema, you need to uncomment the lines referring to `client_info` in the `NIS+LDAPmapping` file to activate the LDAP storage of this information. The schema for `client_info` attributes and object classes in LDIF format is:

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.12.0 \
NAME 'nisplusClientInfoAttr' \
DESC 'NIS+ client_info table client column' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.12.1 \
NAME 'nisplusClientInfoInfo' \
DESC 'NIS+ client_info table info column' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.12.2 \
NAME 'nisplusClientInfoFlags' \
DESC 'NIS+ client_info table flags column' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.42.2.27.5.42.42.13.0 \
NAME 'nisplusClientInfoData' \
DESC 'NIS+ client_info table data' \
SUP top STRUCTURAL MUST ( cn ) \
MAY (nisplusClientInfoAttr $ nisplusClientInfoInfo $ \
nisplusClientInfoFlags))
```

Sample LDIF to create the container.

```
dn: ou=ClientInfo,dc=example,dc=com
ou: ClientInfo
objectClass: top
objectClass: organizationalUnit
```

NIS+ Object Data and Entry Data

The NIS+ naming service stores information about the NIS+ objects themselves, in addition to the entries they contain. This information can be used to set ownership and access rights of the objects. LDAP provides its own mechanism for defining object access rights, so the information is not applicable in LDAP. You might wish to replicate this information to another NIS+ Gateway over LDAP, so storing the information in LDAP might be useful.

The default `NIS+LDAPmapping(4)` file assumes that NIS+ objects will be mapped, so no additional steps are required to activate it. However, as the sample deployment illustrates, you need to comment out lines in `NIS+LDAPmapping` to avoid having to add the additional schema definition. The following is an LDIF representation for adding the NIS+ object schema.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.1.0 \
NAME 'nisplusObject' \
DESC 'An opaque representation of a NIS+ object' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.5 SINGLE-VALUE )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.42.2.27.5.42.42.2.0 \
NAME 'nisplusObjectContainer' \
SUP top STRUCTURAL DESC 'Abstraction of a NIS+ object' \
MUST ( cn $ nisplusObject ) )
```

The following is an LDIF representation that can be imported to create the container.

```
dn: ou=nisPlus,dc=example,dc=com
ou: nisPlus
objectClass: top
objectClass: organizationalUnit
```

Associated with object data is information about ownership, access rights and time to live. The NIS+ Gateway makes assumptions about what this data should be based on the information stored about the object itself. In most cases, it is not necessary to change this behavior.

The following shows the LDIF representation of the `nisplusEntryData` object class and its associated attributes.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.4.0 \
NAME 'nisplusEntryOwner' \
DESC 'Opaque representation of NIS+ entry owner' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.4.1 \
NAME 'nisplusEntryGroup' \
DESC 'Opaque representation of NIS+ entry group' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.4.2 \
NAME 'nisplusEntryAccess' \
DESC 'Opaque representation of NIS+ entry access' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.4.3 \
NAME 'nisplusEntryTtl' \
DESC 'Opaque representation of NIS+ entry TTL' \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.26 SINGLE-VALUE )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.42.2.27.5.42.42.5.0 \
NAME 'nisplusEntryData' \
SUP top STRUCTURAL DESC 'NIS+ entry object non-column data' \
MUST ( cn ) MAY ( nisplusEntryOwner $ nisplusEntryGroup $ \
nisplusEntryAccess $ nisplusEntryTtl ) )
```

Principal Names and Netnames

Principal names and the secure RPC equivalent netnames are used by NIS+ for authentication. While the RFC 2307 specification provides a mapping for public and private keys, there is no provision for storing principal or netnames. This information is maintained in the NIS+ `cred.org_dir` table, for which there is no standard LDAP counterpart.

To get around this issue, the NIS+ Gateway makes some assumptions about how principal names and netnames should be constructed. Basically, the name is constructed by appending the domain name to either the user UID or host nodename. In most cases this works fine. The schema definition for principled names and netnames is shown below.

```
dn: cn=schema
changetype: modify
add: attributetypes
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.7.0 NAME
'nisplusPrincipalName' \
DESC 'NIS+ principal name' \
EQUALITY caseIgnoreIA5Match SINGLE-VALUE \
SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 )
attributetypes: ( 1.3.6.1.4.1.42.2.27.5.42.42.9.0 NAME
'nisplusNetname' \
DESC 'Secure RPC netname' \
EQUALITY caseIgnoreIA5Match SINGLE-VALUE \
YNTAX 1.3.6.1.4.1.1466.115.121.1.15 )

dn: cn=schema
changetype: modify
add: objectclasses
objectclasses: ( 1.3.6.1.4.1.42.2.27.5.42.42.10.0\
NAME 'nisplusAuthName' \
SUP top AUXILLIARY DESC 'NIS+ authentication identifiers' \
MAY ( nisplusPrincipalName $ nisplusNetname ) )
```

NIS+ to LDAP Mapping

The `/var/nis/NIS+LDAPmapping` file controls how NIS+ data is mapped to LDAP attributes and how LDAP attributes are mapped to NIS+ data. At first glance, the syntax of the file can be quite daunting. In practice, however, usually only minor changes are required. In this section, the syntax of the file is examined to give you a better understanding of how the mapping takes place.

The `NIS+LDAPmapping` file contains five directives, which are:

- nisplusLDAPdatabaseIdMapping
- nisplusLDAPentryTtl
- nisplusLDAPobjectDN
- nisplusLDAPattributeFromColumn
- nisplusLDAPcolumnFromAttribute

nisplusLDAPdatabaseIdMapping Directive

This is an identifier the NIS+ Gateway uses establish a relation between an NIS+ object and a label the gateway uses internally. In some cases, the label is used to identify more than one NIS+ object, such as a table object and directory object. All the standard NIS+ objects have IDs assigned. Example:

nisplusLDAPdatabaseIdMapping	passwd:passwd.org_dir
nisplusLDAPdatabaseIdMapping	group:group.org_dir
nisplusLDAPdatabaseIdMapping	auto_master:auto_master.org_dir
nisplusLDAPdatabaseIdMapping	auto_home:auto_home.org_dir
nisplusLDAPdatabaseIdMapping	bootparams:bootparams.org_dir

nisplusLDAPentryTtl Directive

This parameter specifies the time-to-live (TTL) for the `rpc.nisd` cache. There are three values associated with each database ID. The first two values represent a range of time in seconds that `rpc.nisd` randomly picks from when the object is initialized. The third value is used for subsequent cache refreshes. A random time is selected to prevent all objects from refreshing at the same time.

The following are examples:

nisplusLDAPentryTtl	passwd:1800:3600:3600
nisplusLDAPentryTtl	group:1800:3600:3600
nisplusLDAPentryTtl	auto_master:1800:3600:3600
nisplusLDAPentryTtl	auto_home:1800:3600:3600
nisplusLDAPentryTtl	bootparams:1800:3600:3600
nisplusLDAPentryTtl	ethers:1800:3600:3600
nisplusLDAPentryTtl	hosts:1800:3600:3600

nisplusLDAPobjectDN Directive

This parameter specifies where in the DIT NIS+ object data resides. If the DN ends with a trailing comma, the value of the `defaultSearchBase` parameter (set in the `rpc.nisd` configuration) is appended to it. There are three action fields that determine:

- Where to read data from
- Where to write data to
- What to do if the LDAP data is deleted

If the third action field is left out, the default action is to delete the entire entry. Below is an example.

Example:

```
nisplusLDAPobjectDN    hosts:ou=Hosts,?one?objectClass=ipHost:\
ou=Hosts,?one?objectClass=ipHost,\
objectClass=device,objectClass=top
```

Note – Because where to read the data from and where to write it to is the same, the write location can be omitted as long as the colon is present.

The action is shown as an LDAP search filter. The

`hosts:ou=Hosts,?one?objectClass=ipHost` is read as follows (assuming the domain is `example.com`):

1. Start the search as `ou=Hosts,dc=example,dc=com`
2. Perform a one-level search (do not search below `ou=Hosts`).
3. Search only for entries that contain the object class of `ipHost`.

The write action is the same as the read. The delete action is the default. That is, delete the entire entry.

nisplusLDAPAttributeFromColumn Directive

This directive is used to specify rules for mapping NIS+ data to LDAP. For example:

```
nisplusLDAPAttributeFromColumn \  
hosts: dn=("cn=%s+ipHostNumber=%s,", cname, addr), \  
cn=cname, \  
cn=name, \  
ipHostNumber=addr, \  
description=comment
```

In this example, the relative distinguished name (RDN) of the LDAP entry would be the `cn=` attribute with the value of the NIS+ `cname` (canonical name) attribute, followed by the string `+ipHostNumber=`, then the value of the NIS+ `addr` attribute. The entry contains a `cn` attribute with the value of `cname` and if the host name has an alias, another `cn` attribute with the alias name. The IP address is assigned to the `ipHostNumber` attribute and if a comment field exists, the value of it is assigned to the `description` attribute. For example, if the NIS+ entry looks like this:

```
# niscat -h hosts.org_dir  
# cname name addr comment  
localhost localhost 127.0.0.1  
esso mailserver 10.10.9.12 #Mail server for Bldg 1  
.  
.  
.
```

The corresponding LDAP entry looks like this:

```
dn: cn=esso+ipHostNumber=10.10.9.12,\  
ou=Hosts,dc=example,dc=com  
objectClass: ipHost  
objectClass: device  
objectClass: top  
cn: esso  
cn: mailserver  
ipHostNumber: 10.10.9.12  
description:Mail server for Bldg 1
```

`nisplusLDAPcolumnFromAttribute` Directive

This directive performs the opposite action of the `nisplusLDAPattributeFromColumn` directive. That is, it defines the rules to map LDAP data to NIS+. For example:

```
nisplusLDAPcolumnFromAttribute \  
hosts: cname=cn, \  
(name)=(cn), \  
addr=ipHostNumber, \  
comment=description
```

Using the Default Configuration Files

In most cases, you will want to change some of the default configuration parameters. However, you can run the NIS+ Gateway without any modifications to the configuration files. A list of assumptions is presented here to help you gauge how much modification will be required to fit your environment.

Assumptions

- The directory server is running on the same server as the NIS+ Gateway. That is, the server IP address is set to 127.0.0.1 and the port number is 389.
- The NIS+ standard directories and tables are to be mapped, with the exception of `timezone.org_dir` and `client_info.org_dir`. This implies that the schema to hold the NIS+ table objects (`nisObjectContainer`) needs to be defined on the directory server. The schema definitions for `timezone` and `client_info` do not need to be defined.
- The mapping is two way—that is, from NIS+ to LDAP and from LDAP to NIS+. This configuration allows updates to be made from either service.
- The authentication method defined is `none`. This implies anonymous access to the directory and there is no transport layer security.
- The directory server supports the RFC2307bis schema. This is the default configuration for the Sun ONE Directory Server software.
- Only the entry values are mapped. Information contained in pseudo-columns such as table owner and group is not mapped.
- The default NIS+ credentials are sufficient.
- The configuration data will not be maintained in LDAP.

Common Configuration Changes

The following is a list of changes that you probably will want to make.

1. Directory Server address and port number.

In the `rpc.nisd` file, change: `preferredServerList=127.0.0.1:389`

2. Authentication Method:

`nisplusLDAPproxyUser`

3. The bind-DN to use for authentication:

`nisplusLDAPproxyPassword`

4. Transport Security:

To use transport layer security, set `nisplusLDAPconfigTLS` or `nisplusLDAPTLS` to `ssl`, and set `nisplusLDAPconfigTLSCertificateDBPath` or `nisplusLDAPTLSCertificateDBPath` to the file containing the certificate database. In order to successfully use authentication and transport layer security, the server must also support the chosen values.

5. Time-to-Live Parameters

To change the TTLs, edit the `nisplusLDAPentryTtl` for the appropriate database ID. To change LDAP container locations or object classes, edit the `nisplusLDAPobjectDN` value for the appropriate database ID.

6. NIS+ Table Object Support

The `nisplusObjectContainer`, `nisplusObject`, and `ou=nisPlus` labels are suggestions. If you change `nisplusObjectContainer`, or `ou=nisPlus`, edit the mapping file to reflect this. To change `nisplusObject`, for example, to `myObject`, add `nisplusObject=myObject` to the `filterAttrValList` and `attrValList` portions of the `readObjectSpec` and `writeObjectSpec` of the `nisplusLDAPobjectDN` value for the mapping. See the description of `nisplusLDAPobjectDN` below.

NIS+ to LDAP Migration Example

The NIS+ Gateway is very flexible, providing many different ways to configure it to meet your needs. Providing examples that cover all the different possible configurations would be very long and would be more confusing than beneficial. Therefore, the following assumptions are made about the environment where the NIS+ Gateway is run.

- The target directory server is Sun ONE Directory Server 5.x software.
- The `idsconfig` script is run on the directory server.
- The directory server is set up to store passwords in crypt format.
- The current NIS+ service consists of a single domain with one NIS+ master server.
- The `timezone` and `client_info` tables are not mapped. This eliminates the need to update the directory server schema.
- Table objects are not mapped. This eliminates the need to update the directory server schema.
- The LDAP structure is empty and will be populated by running `rpc.nisd`.

▼ To Migrate Your Data From NIS+ to LDAP

Before proceeding, make sure you backup all NIS+ data; see `nisbackup(1M)`.

1. **Upgrade your NIS+ master server to run the Solaris 9 OE.**
2. **Install the Sun ONE Directory Server software, and run the `idsconfig` script.**

The sample deployment assumes the Sun ONE Directory Server software is running on the same system as the NIS+ Gateway, but this is not necessary. If you choose not to run them on the same system, the server IP address and port number must be defined in `rpc.nisd`.

3. **Obtain the password for the `cn=directory manager` account.**

The sample deployment uses this account as the proxy user for accessing data stored in the LDAP directory. For a real production environment you should to create a separate account for this purpose and grant that account read and write access privileges.

4. **Edit the following lines in `/etc/default/rpc.nisd`.**

- `preferredServerList=127.0.0.1:389` – You do not have to edit this line if the directory server is running on the same system as NIS+ Gateway.

- `defaultSearchBase=dc=example,dc=com` - You do not have to edit this line if the directory server DIT equates to your NIS+ domain name.
- `authenticationMethod=simple`
- `nisplusLDAPproxyUser=cn=directory manager` - If you choose not to use this account, specify the DN of an existing account.
- `nisplusLDAPproxyPassword=dirmanager` - Make sure you read protect the `rpc.nisd` file to protect the password.

5. Create a copy of the `/var/nis/NIS+LDAPmapping.template` file.

The copy should be created in the `/var/nis` directory and can be called anything except `NIS+LDAPmapping`. For this example, we assume the copy of the mapping file is `/var/nis/nlm`.

6. Edit the `/var/nis/nlm` file to disable directory and group objects.

```
# Standard NIS+ directories
#nisplusLDAPdatabaseIdMapping basedir:
#nisplusLDAPdatabaseIdMapping orgdir:org_dir
#nisplusLDAPdatabaseIdMapping groupsdir:groups_dir

# Standard NIS+ groups.
#nisplusLDAPdatabaseIdMapping admin:admin.groups_dir

# Standard NIS+ directories
#nisplusLDAPentryTtl basedir:21600:43200:43200
#nisplusLDAPentryTtl orgdir:21600:43200:43200
#nisplusLDAPentryTtl groupsdir:21600:43200:43200
#nisplusLDAPentryTtl admin:21600:43200:43200

# Standard NIS+ directories
#nisplusLDAPobjectDN basedir:cn=basedir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer:\
# cn=basedir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer,\
# objectClass=top

#nisplusLDAPobjectDN orgdir:cn=orgdir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer:\
# cn=orgdir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer,\
# objectClass=top

#nisplusLDAPobjectDN
#groupsdir:cn=groupsdir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer:\
# cn=groupsdir,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer,\
# objectClass=top

#nisplusLDAPobjectDN admin:cn=admin,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer:\
# cn=admin,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer,\
# objectClass=top
```

7. Edit the /var/nis/nlm file to remove all references to table objects.

Any entry containing `_table` needs to be commented out. Example (not all entries shown for brevity):

```
#nisplusLDAPdatabaseIdMapping passwd_table:passwd.org_dir
#nisplusLDAPdatabaseIdMapping group_table:group.org_dir
#nisplusLDAPdatabaseIdMapping
auto_master_table:auto_master.org_dir
.
.
.
nisplusLDAPentryTtl passwd_table:21600:43200:43200
#nisplusLDAPentryTtl group_table:21600:43200:43200
#nisplusLDAPentryTtl auto_master_table:21600:43200:43200
.
.
.
#nisplusLDAPobjectDN
#passwd_table:cn=passwd,ou=nisPlus,?base?\
#objectClass=nisplusObjectContainer:\
# cn=passwd,ou=nisPlus,?base?\
# objectClass=nisplusObjectContainer,\
# objectClass=top
.
.
.
```

8. Test the mapping you created using the `nislmaptest` utility with the `im` option specifying the mapping file `nlm`.

Note – Make sure you remove the test data entries before performing the next step.

9. Upload all NIS+ data to LDAP using `rpc.nisd`.

Example:

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

10. Verify that the LDAP entries were created.

See the instructions in the next section that explain how to perform the verification.

11. **Rename** `/var/nis/nlm` to `/var/nis/NIS+LDAPmapping` and restart `rpc.nisd`.

```
# cd /var/nis
# cp nlm NIS+LDAPmapping
# /usr/sbin/rpc.nisd
```

Testing and Troubleshooting the NIS+ Gateway

Before uploading all your NIS+ data to the LDAP server, it is wise to test out the mapping you established. This should be done before you rename the configuration file to `NIS+LDAPmapping` and restart `rpc.nisd`, which activates the NIS+ Gateway.

The following assumes you have a group in your NIS+ database called `gem`. The `nisldapmaptest` command is run in the verbose mode to give you a clearer idea of what the command is doing.

```
# nisldapmaptest -v -r -m nls -t group.org_dir name=gem
createQuery: NIS+ query: [name=gem]group.org_dir.example.com.
mapToLDAP: group.org_dir.example.com.: 1 * 1 potential updates
mapToLDAP: group.org_dir.example.com.: 1 update requested
controlSupported: 127.0.0.1: 1.2.840.113556.1.4.319: disabled
controlSupported: 127.0.0.1: 2.16.840.1.113730.3.4.9: enabled
mapToLDAP: group.org_dir.example.com.: 1 update performed
```

As you can observe from the last line the update was performed. To verify this, you can run the `ldapsearch` command as shown in the following example.

```
# ldapsearch -b dc=example,dc=com cn=gem
cn=gem,ou=Group,dc=example,dc=com
cn=gem
gidNumber=2292
objectClass=posixGroup
objectClass=top
```

To remove test entries:

```
# ldapsearch -b dc=example,dc=com cn=gem dn
cn=gem,ou=Group,dc=example,dc=com
# ldapdelete -D "cn=directory manager" -w mypass
cn=gem,ou=Group,dc=example,dc=com
```

Once the NIS+ Gateway is activated, you can test it by creating an entry in LDAP and see if it appears in NIS+.

```
# vi /tmp/hosts.ldif
dn: cn=test1+ipHostNumber=10.10.9.12,ou=Hosts,dc=example,dc=com
cn: test1
ipHostNumber: 10.10.9.12
objectClass: ipHost
objectClass: device
objectClass: top

# ldapmodify -a -D "cn=directory manager" -f /tmp/hosts.ldif
Bind Password:
adding new entry
cn=test1+ipHostNumber=10.10.9.12,ou=Hosts,dc=example,dc=com
#
```

After a few seconds, you should be able to see the new entry from NIS+ as shown in this example.

```
# niscat hosts.org_dir | grep test1
test1 test1 10.10.9.12
```

Troubleshooting Tips

Most errors are the result of misconfigured `rpc.nisd` or `NIS+LDAPmapping` files. If you encounter problems, follow these steps:

- Check for syslog messages in `/var/adm/messages` that are generated from `rpc.nisd`.
- Restart `rpc.nisd` with the verbose (`-v`) option for more detailed error messages.
- Run the `nislmaptest(1M)` utility for entries that fail, observing error messages.

- Run `snoop` if the directory server is on a separate system. See Appendix C, “Using `snoop` with LDAP.”

Common Problems

- The directory server is missing necessary schema definitions. This will occur if you fail to comment out the `directory`, `group`, and `table` entries in `NIS+LDAPmapping(4)` file and do not add the `nisObjectContainer` object class and associated attributes.
- The naming service containers, for example `ou=hosts`, have not been created. To avoid this, run the `idsconfig` command before uploading NIS+ data.
- The proxy user account specified does not have sufficient write permissions. For test purposes you can choose `cn=directory manager` to avoid this issue.
- The NIS+ tables being imported have duplicate entries. While this is allowed in NIS+, you will get an LDAP object class violation error when they are imported because it will be interpreted as a multi-value attribute.

About the Authors

Michael Haines is a principal staff engineer for Sun Microsystems, Inc. He started his career in the CTE engineering group, and has been at Sun almost 14 years. Michael has held various engineering positions within Sun, and is the coauthor of the Sun BluePrints “Solaris and LDAP Naming Services – Deploying LDAP in the Enterprise,” published in 2001.

Tom Bialaski is a senior staff engineer in the Enterprise Engineering group at Sun Microsystems. He began his career at Sun as a Systems Engineer almost 20 years ago, and has held various customer-focused engineering positions since then. Tom is the coauthor of the Sun BluePrints “Solaris and LDAP Naming Services – Deploying LDAP in the Enterprise,” published in 2001.

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`