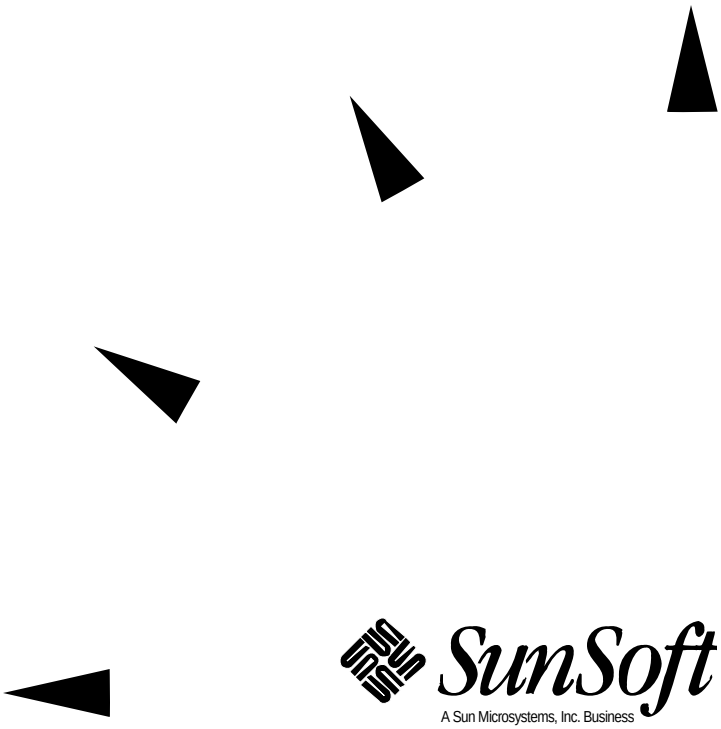
# NIS+ Transition Guide



2550 Garcia Avenue
Mountain View, CA 94043
U.S.A.

SunSoft
A Sun Microsystems, Inc. Business

# *Contents*

# *Figures*

# *Tables*

# *Preface*

*NIS+ Transition Guide* describes how to convert a site running the NIS name service to one running the NIS+ name service. This manual is part of the Solaris™ 2.x System and Network Administration set.

## *Who Should Use This Book*

*NIS+ Transition Guide* is for experienced system and network administrators who want to convert their sites from NIS to NIS+. For information on configuring NIS+, see *NIS+ and DNS Setup and Configuration Guide*. For NIS+ customizing information and detailed administration instructions, see *NIS+ and FNS Administration Guide.*

Although this manual introduces some networking concepts relevant to NIS+, it makes no attempt to explain networking fundamentals or describe the administration tools offered by the Solaris environment. If you administer networks, you should already know how the administration tools work and have already chosen your favorite tools.

## *How This Book Is Organized*

This book contains six chapters.

**Chapter 1, "Introduction**," describes the differences between NIS and NIS+ features and an overview of the suggested transition process.

**Chapter 2, "Designing the NIS+ Namespace**," discusses how to design your NIS+ namespace.

**Chapter 3, "Planning NIS+ Security Measures**," describes the NIS+ security features and the effects they have on administration and transition planning.

**Chapter 4, "Using NIS-Compatibility Mode**," describes how to run NIS and NIS+ clients concurrently and NIS+ servers in NIS-compatibility mode.

**Chapter 5, "Prerequisites to Transition**," presents the steps that you need to take before beginning the actual transition.

**Chapter 6, "Implementing the Transition**," lists the steps required to implement an NIS-to-NIS+ transition.

## *Related Books*

Consult the following publications for more information on NIS+ and DNS:

- *NIS+ and DNS Setup and Configuration Guide*—Describes how to plan for, set up, and configure an NIS+ namespace
- *NIS+ and FNS Administration Guide*—Describes how to administer a running NIS+ namespace and modify its security level

## What Typographic Changes Mean

The following table describes the typographic changes used in this book.

*Table P-1* Typographic Conventions

| Typeface or Symbol | Meaning | Example |
|---|---|---|
| `AaBbCc123` | The names of commands, files, and directories; on-screen computer output | Edit your `.login` file.<br>Use `ls -a` to list all files.<br>`machine_name% You have mail.` |
| **`AaBbCc123`** | What you type, contrasted with on-screen computer output | `machine_name% su`<br>`Password:` |
| *AaBbCc123* | Command-line placeholder: replace with a real name or value | To delete a file, type `rm` *filename*. |
| *AaBbCc123* | Book titles, new words or terms, or words to be emphasized | Read Chapter 6 in *User's Guide*.<br>These are called *class* options.<br>You *must* be root to do this. |

## Shell Prompts in Command Examples

The following table shows the default system prompt and superuser prompt for the C shell, Bourne shell, and Korn shell.

*Table P-2* Shell Prompts

| Shell | Prompt |
|---|---|
| C shell prompt | `machine_name%` |
| C shell superuser prompt | `machine_name#` |
| Bourne shell and Korn shell prompt | `$` |
| Bourne shell and Korn shell superuser prompt | # |

# Introduction 1

This chapter introduces the issues involved in converting from NIS to NIS+. It describes the differences between the two name services and outlines a suggested transition process.



## Differences Between NIS and NIS+

NIS and NIS+ have several differences that have an impact on a transition. For example, NIS uses a flat, non-hierarchical namespace with only one domain (or several disconnected domains), while NIS+ provides a domain hierarchy similar to that of DNS. This means that before you can convert to NIS+, you must design the NIS+ namespace. NIS+ also provides security, which limits access not only to the information in the namespace but also to the structural components of the namespace.

These and other differences demonstrate that NIS+ is not simply an upgrade to NIS but an entirely new product. Therefore, the transition from NIS to NIS+ is largely directed by the differences between the products.

These differences are described in broad terms in the remainder of this chapter. Understanding them is critical to a successful transition to NIS+. They are:

- Domain structure
- Interoperability

- Server configuration
- Information management
- Security

## Domain Structure

NIS+ is not simply an upgrade to NIS; it is designed to *replace* NIS. This becomes evident when you examine its domain structure. NIS domains are flat and lack the ability to have a hierarchy. NIS+ domains *may* be flat, but you can also construct hierarchical NIS+ domains. Such hierarchies consist of a root domain with an infinite number of subdomains under them.

The NIS domain structure addressed the administration requirements of client-server computing networks prevalent in the 1980s; in other words, client-server networks with a few hundred clients and a few multipurpose servers.

NIS+ is designed to support networks with 100 to 10,000 multivendor clients supported by 10 to 100 specialized servers located in sites throughout the world, connected to several “untrusted” public networks. The size and complexity of these networks requires new, autonomous administration practices. The NIS+ domain structure was designed to address these requirements. It consists of hierarchical domains similar to those of DNS, as shown in the following diagram:

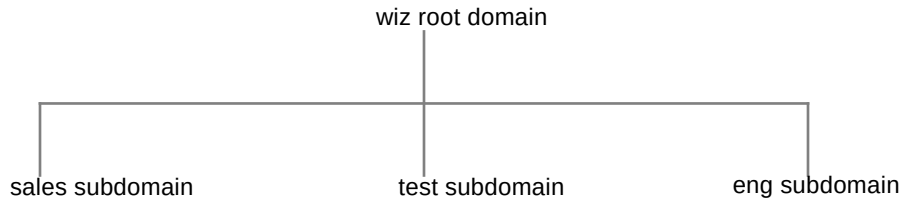



*Figure 1-1* NIS+ Domains

Hierarchical domains allow NIS+ to be used in a range of networks, from small to very large. They also allow the NIS+ service to adapt to the growth of an organization. The NIS+ domain structure is thoroughly described in *NIS+ and FNS Administration Guide.*

## *DNS, NIS, and NIS+ Interoperability*

NIS+ provides interoperability features designed for upgrading from NIS and for continuing the interaction with DNS originally provided by the NIS service.

To help convert from NIS, NIS+ provides an NIS-compatibility mode and an information transfer utility. The NIS-compatibility mode enables an NIS+ server running Solaris 2.x software to answer requests from NIS clients while continuing to answer requests from NIS+ clients. The information transfer utility helps administrators keep NIS maps and NIS+ tables synchronized.

NIS-compatibility mode requires slightly different setup procedures than those used for a standard NIS+ server. Also, NIS-compatibility mode has security implications for tables in the NIS+ namespace. These differences and implications are described in *NIS+ and DNS Setup and Configuration Guide* and *NIS+ and FNS Administration Guide.*

NIS client machines interact with the NIS+ namespace differently from NIS+ client machines when NIS+ servers are running in NIS-compatibility mode. The differences are:

- NIS client machines cannot follow NIS+ table paths or links, or do read operations in other domains.
- NIS client machines can have their unsatisfied host requests forwarded to DNS if you run `rpc.nisd` with the `-Y -B` options, but the NIS+ server will not forward these requests for an NIS+ client. DNS request forwarding for NIS+ client machines is controlled by the `resolv.conf` and `nsswitch.conf` files' configurations. See *NIS+ and FNS Administration Guide* for more information.
- Authorized NIS+ administrators can use the `passwd` command to perform the full range of password-related administrative tasks including password aging and locking. NIS+ client users can use the `password` command to change their own passwords.
- Even if all the servers on a local subnet no longer respond, the NIS+ client machines can still have their name service calls answered if they can contact any of the replicas of that domain. NIS client machines do not have access to information on the network outside their subnet unless the server names have been set with `ypset` or, for Solaris 2.x NIS clients only, with `ypinit`.

- NIS client machines cannot be sure that the data they are receiving comes from an authorized NIS server, while authorized NIS+ clients are certain that the data is coming from an authorized NIS+ server.
- Under NIS, if the server is no longer responding, the NIS `yp_match()` call continues to retry this call until the server starts responding and answers the request. The NIS+ API (application programing interface) will return an error message to the application when this situation occurs.

In the Solaris 2.3 and later releases, the NIS-compatibility mode *supports* DNS forwarding. In the Solaris 2.2 release, support for DNS forwarding is available as a *patch*. The DNS forwarding patch is *not* available in the Solaris 2.0 and 2.1 releases.

Although an NIS+ domain cannot be connected to the Internet directly, the NIS+ client machines can be connected to the Internet with the name service switch. The client can set up its switch-configuration file (`/etc/nsswitch.conf`) to search for information in either DNS zone files or NIS maps—in addition to NIS+ tables.

## *Server Configuration*

The NIS+ client-server arrangement is similar to those of NIS and DNS in that each domain is supported by a set of servers. The main server is called the *master* server, and the backup servers are called *replicas*. Both master and replica servers run NIS+ server software and both maintain copies of NIS+ tables.

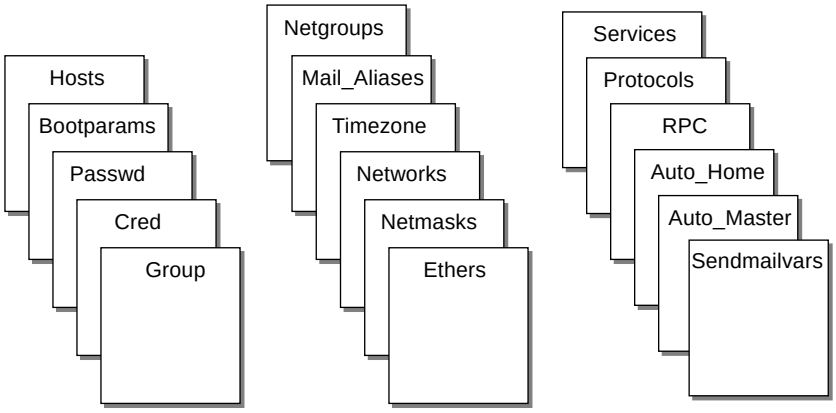
However, NIS+ uses an update model that is completely different from the one used by NIS. At the time NIS was developed, it was assumed that most of the information NIS would store would be static. NIS updates are handled manually, and its maps have to be remade and propagated in full every time any information in the map changes.

NIS+, however, accepts *incremental* updates to the replicas. Changes must still be made to the master database on the master server, but once made they are automatically propagated to the replica servers. You don’t have to “make” any maps or wait hours for propagation. Propagation now takes only a matter of minutes.

## *Information Management*

NIS+ stores information in *tables* instead of maps or zone files. NIS+ provides 17 types of predefined, or *system*, tables, as shown in Figure 1-2:

*Figure 1-2* NIS+ Standard Tables




NIS+ tables are not ASCII files, but are tables in the NIS+ relational database. You can view and edit their contents only by using the NIS+ commands.

NIS+ tables provide two major improvements over the maps used by NIS. First, an NIS+ table can be searched by any searchable column, not just the first column (sometimes referred to as the “key”). To know whether a particular column is searchable, run the `niscat -o` command on a table. The command returns a list of the table’s columns and their attributes, one of which is whether a column is searchable. This search ability eliminates the need for duplicate maps, such as the `hosts.byname` and `hosts.byaddr` maps used by NIS. Second, the information in NIS+ tables has access controls at three levels: the table level, the entry (row) level, and the column level.

NIS maps are located on the server in `/var/yp/`*domainname*, whereas NIS+ directories are located in `/var/nis/data`. The NIS+ tables are contained in the database. The tables’ information is loaded into memory as requests are made to the database. Keeping data in memory in the order requested minimizes calls to the disk, thereby improving request response time.

# 1

### *Security*

The security features of NIS+ protect the information in the namespace and the structure of the namespace itself from unauthorized access. NIS+ security is provided by two means: *authentication* and *authorization*. Authentication is the process by which an NIS+ server identifies the NIS+ *principal* (a client user or client workstation) that sent a particular request. Authorization is the process by which a server identifies the access rights granted to that principal, whether a client machine or client user.

In other words, before users can access anything in the namespace, they must be authenticated NIS+ clients and they must have the proper permission to access that information. Furthermore, requests for access to the namespace are only honored if they are made either through NIS+ client library routines or NIS+ administration commands. The NIS+ tables and structures cannot be edited directly.

## *Suggested Transition Phases*

The following outline is a suggested NIS-to-NIS+ transition:

1. Review basic transition principles.
2. Become familiar with NIS+.
3. Design your final NIS+ namespace.
4. Select security measures.
5. Decide how to use NIS-compatibility mode.
6. Complete prerequisites to transition.
7. Implement the transition.

The remainder of this chapter is a detailed discussion of these transition phases.

### *Transition Principles*

Before you begin the transition, you should review the following basic principles:

### *Consider the Alternatives to Making the Transition Immediately*

You can defer the upgrade to NIS+ until after your site has completed its transition to the Solaris 2.x release. This would allow you to focus your resources on one transition effort at a time. Use the binary-compatibility mode Name Service kit to continue running NIS after you upgrade to the Solaris 2.x release. It is a much better solution than running NIS+ with the Solaris 1.x release.

### *Keep Things Simple*

You can take several steps to simplify the transition. While these steps will diminish the effectiveness of NIS+, they will consume fewer servers and less administrative time. Once the transition is complete, you can change the NIS+ setup to better suit your needs. Here are some suggestions:

- Don’t change domain names.
- Don’t use any hierarchies; keep a flat NIS+ namespace.
- Use the NIS-compatibility features.
- Use default tables and directory structures.
- Don’t establish credentials for clients.

### *Use a Single Release of Software*

Decide which version of the Solaris 2.x software and NIS+ you will use for the transition. Since there are slight differences between versions, using multiple versions could needlessly complicate the transition process. Choose one version of the Solaris product and use its corresponding version of NIS+.

The current release has the most features (such as setup scripts). Make sure you compile a list of the SunOS patches that are required for normal operation, and make sure that all servers and clients have the same patches loaded.

### *Minimize Impact on Client Users*

There are two major user-related considerations. First, users should not notice any change in service. Second, the transition phase itself should cause minimal disruption to client users. To ensure the second consideration, be sure the administrators responsible for each domain migrate their client machines to NIS+, rather than ask the users to implement the migration. This ensures that proper procedures are implemented, that procedures are consistent across client machines, and that irregularities can be dealt with immediately by the administrator.

***Don't—***

- Don't change the name services currently provided by NIS or the way NIS functions.
- Don't change the structure of DNS.
- Don't change the IP network topology.
- Don't upgrade applications that use NIS to NIS+; leave the migration to NIS+ APIs for the future.
- Don't consider additional uses for NIS+ during the implementation phase; add them later.

## *Become Familiar with NIS+*

Familiarize yourself with NIS+, particularly with the concepts summarized earlier in this chapter and discussed in the remainder of this book. For details, see the publications listed in "Related Books" on page xii.

One of the best ways to become familiar with NIS+ is to build a prototype namespace. There is no substitute for hands-on experience with the product; administrators need the opportunity to practice in a forgiving test environment.

---

**Note –** Do not use your prototype domain as the basis for your actual running NIS+ namespace. Deleting your prototype when you have learned all you can from it will avoid namespace configuration problems. Start anew to create the real namespace after following all the planning steps.

---

When you create the test domains, make small, manageable domains. For guidance, you can use *NIS+ and DNS Setup and Configuration Guide*, which describes how to plan and create a simple test domain and subdomain (with or without NIS-compatibility mode) using the NIS+ set up scripts.

---

**Note –** The NIS+ scripts described in Part—1 of *NIS+ and DNS Setup and Configuration Guide* are the recommended method of setting up an NIS+ namespace. The recommended procedure is to first set up your basic NIS+ namespace using the scripts, and then customize that namespace for your particular needs using the NIS+ command set.

---

## *Design Your Final NIS+ Namespace*

Design the final NIS+ namespace, following the guidelines in Chapter 2, “Designing the NIS+ Namespace.” While designing the namespace, do not worry about limitations imposed by the transition from NIS. You can add those later, once you know what your final NIS+ goal is.

## *Plan Security Measures*

NIS+ security measures provide a great benefit to users and administrators, but they require additional knowledge and setup steps on the part of both users and administrators. They also require several planning decisions. Chapter 3, “Planning NIS+ Security Measures,” describes the implications of NIS+ security and the decisions you need to make for using it in your NIS+ namespace.

## *Decide How to Use NIS-Compatibility Mode*

The use of parallel NIS and NIS+ namespaces is virtually unavoidable during a transition. Because of the additional resources required for parallel namespaces, try to develop a transition sequence that reduces the amount of time your site uses dual services or the extent of dual services within the namespace (for example, convert as many domains as possible to NIS+ only).

Chapter 4, “Using NIS-Compatibility Mode,” explains the transition issues associated with the NIS-compatibility mode and suggests a way to make the transition from NIS, through NIS compatibility, to NIS+ alone.

## *Complete Prerequisites to Transition*

In addition to the planning decisions mentioned above, you must complete several miscellaneous prerequisites, as described in Chapter 5, “Prerequisites to Transition.”

## *Implement the Transition*

Chapter 6, “Implementing the Transition” provides suggested steps to implement the transition you have planned in the previous steps.

# 1

# Designing the NIS+ Namespace 2

This chapter provides general guidelines and recommendations for designing the final NIS+ namespace your site will have.



When designing the namespace, do not worry about limitations imposed by the transition from NIS. You can modify your NIS+ domain later, once you know what your final NIS+ configuration will look like.

## Identifying the Goals of Your Administrative Model

Select the model of information administration, such as the domain structure, that your site will use. Without a clear idea of how information at your site will be created, stored, used, and administered, it is difficult to make the design decisions suggested in this section. You could end up with a design that is more expensive to operate than necessary. You also run the risk of designing a namespace that does not suit your needs. Changing the namespace design after it has been set up is costly.

# 2

# Designing the Namespace Structure

Designing the NIS+ namespace is one of the most important tasks you can perform, since changing the domain structure after NIS+ has been set up is a time-consuming, complex job. It is complex because information, security, and administration policies are woven into the domain structure of the namespace. Rearranging domains would require rearranging information, reestablishing security, and recreating administration policies.

When designing the structure of an NIS+ namespace, consider the following factors which are discussed in the following sections of this chapter:

- "The Domain Hierarchy"
- "Domain Names"
- "The Email Environment"

## The Domain Hierarchy

The main benefit of an NIS+ domain hierarchy is that it allows the namespace to be divided into more easily managed components. Each component can have its own security, information management, and administration policies. It is advisable to have a hierarchy if the number of clients you have exceeds 500, if you want to set up different security policies for a set of users, or if you have geographically distributed sites.

Unless there is a need for a domain hierarchy, not having a hierarchy will simplify your transition to NIS+ because the NIS+ servers for each subdomain are not part of the subdomain that they serve, with the exception of the root domain. The NIS+ servers are in the parent domain of the subdomain they serve. This relationship of server to subdomain creates problems for applications that expect the servers to be able to get their name service data from the subdomain. For example, if a subdomain NIS+ server is also an NFS server, then the server would not get its netgroups information from the subdomain, but instead retrieve the information from its domain, which is the domain above the subdomain; this can be confusing. Another example of when a hierarchy could cause problems would be where the NIS+ server was also used by users to log in remotely and to execute certain commands that they could not execute from their own workstations. If you have only a single root domain, you will not have these problems because NIS+ root servers live in the domain that they serve.

When all users were in the same NIS domain, they were directly visible to each other without using fully qualified names. Creating an NIS+ hierarchy, however, puts users in separate domains, which means that the users in one domain will not be directly visible to users in another domain unless you use fully qualified names or paths.

For example, if there are two subdomains, sales.wiz.com. and factory.wiz.com., created out of the earlier wiz.com. domain, then for user “juan” in the sales.wiz.com. domain to be able to send mail to user “myoko” in factory.wiz.com., he would have to specify her name as `myoko@hostname.factory.wiz.com.` (or `myoko@hostname.factory`) instead of just `myoko`, as was sufficient when they were in the same domain. Remote logins also require fully qualified names between domains.

You could use the table path to set up connections between tables in one domain and another domain, but to do so would negate the advantages of having a domain hierarchy. You would also be reducing the reliability of the NIS+ service because now clients would have to depend upon the availability of not only their own home domains, but also of other domains to which their tables are pathed. Using table paths may also slow request response time.

## *Designing a Domain Hierarchy*

If you are unfamiliar with domain hierarchies, first read Part I of *NIS+ and FNS Administration Guide*. It describes NIS+ domain structure, information storage, and security.

Once you are familiar with the components of a domain hierarchy, make a diagram of how you expect the hierarchy to look when you are finished. The diagram will be a useful reference when you are in the midst of the setup procedure. At a minimum, you will need to consider the following issues:

- Organizational or geographical mapping
- Connection to higher domain
- Client support in the root domain
- Domain size compared to number of domains
- Number of levels
- Security levels
- Replicas and number of replicas
- Information management

### *Organizational or Geographical Mapping*

One of the major benefits of NIS+ is its capability of dividing the namespace into smaller, manageable parts. You could create a hierarchy of organizations, such as those of the hypothetical corporation, Wizard, Inc.

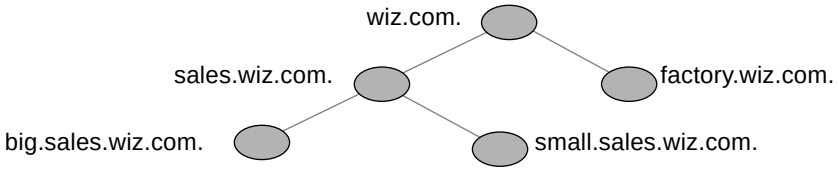



*Figure 2-1* Sample NIS+ Hierarchy by Logical Organization

You could also organize the hierarchy by buildings instead of organizations:

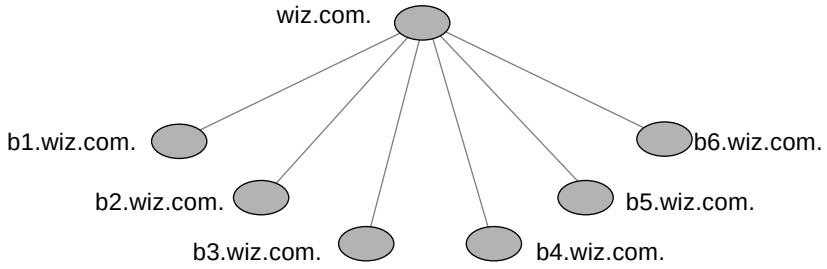



*Figure 2-2* Sample NIS+ Hierarchy by Physical Location

The scheme you select depends primarily on how you prefer to administer the namespace and how clients will tend to use the namespace. For example, if clients of factory.wiz.com. will be distributed throughout the buildings of Wizard, Inc., you should not organize the namespace by building. Since the clients would constantly need to have access to other domains, you would need to add their credentials to the other domains and you would increase traffic flow through the root master server. A better scheme would be to arrange clients by organization. On the other hand, building-sized domains are immune to the reorganizations that require organization-based domains to be restructured.

Do not be limited by the physical layout of the network, since an NIS+ namespace does not have to be congruent with the physical network, except where it has to support NIS clients. The number of domains your namespace needs depends on the kind of hierarchy you select.

Consider future expansion plans. Will today's NIS+ root domain be beneath another NIS+ domain in the future? Changing this arrangement would entail a great deal of work. Try to estimate the need for future domains in the namespace and design a structure that can accommodate them without disruption.

### *Connection to Higher Domains?*

Consider whether the NIS+ namespace will be connected to higher domains, such as those of the Internet or DNS. If you currently use NIS under a DNS hierarchy, do you want to replace only the NIS domains or do you want to replace the entire companywide DNS/NIS structure with an NIS+ namespace?

### *Client Support in the Root Domain*

In the two Wizard, Inc., domain hierarchies illustrated in Figure 2-1 and Figure 2-2 on page 14, are all the clients placed in domains beneath the root domain? Or do some belong to the root domain? Is the purpose of the root domain to act only as the root for its subdomains or will it support its own group of clients? You could place all clients in the lowest layer of domains and only those used for administration in the root and any intermediate domains. For example, if you implemented the plan in Figure 2-1, all clients would belong to the big.sales.wiz.com., small.sales.wiz.com., and factory.wiz.com. domains, and only clients used for administration would belong to the wiz.com. and sales.wiz.com. domains.

Or you could place the clients of general-purpose departments in higher-level domains. For example, in Figure 2-2, where the domain is organized by building, you could put the clients of the Facilities Department in the wiz.com. domain. It is not recommended that you do so, however, because the root domain should be kept simple and relatively unpopulated.

### *Domain Size Compared With Number of Domains*

The current NIS+ implementation is optimized for up to 1000 NIS+ clients per domain and for up to 10 replicas per domain. Such a domain would typically have 10,000 table entries. The limitations come from the current server discovery protocol. If you have more than 1000 NIS+ clients, you should divide your namespace into different domains and create a hierarchy.

Creating a hierarchy, however, may introduce more complexity than you are prepared to handle. You may still prefer to create larger domains rather than a hierarchy because one large domain requires less administration than multiple smaller domains do. Larger domains need fewer skilled administrators to service them, since tasks can be automated more readily (with scripts you create), thus lowering the administrative expense. Smaller domains provide better performance, and you can customize their tables more easily. You also achieve greater administrative flexibility with smaller domains.

### *Number of Levels*

NIS+ was designed to handle multiple levels of domains. Although the software can accommodate almost any number of levels, a hierarchy with too many levels would be difficult to administer. For example, the names of objects could become long and unwieldy. Consider 20 to be the limit for the number of subdomains for any one domain and limit the levels of the NIS+ hierarchy to 5.

### *Security Level*

Typically, you will run the namespace at security level 2. However, if you plan to use different security levels for different domains, you should identify them now. Chapter 3, “Planning NIS+ Security Measures,” provides more information about security levels.

### *Replicas and Number of Replicas*

Any one domain should have no more than 10 replicas because of the increased network traffic and server load that occur when information updates are propagated to the replicas. Determining the number of replicas a domain requires depends on other factors as well. They are:

- The physical location of the servers
- The number of subnets in a domain
- Whether there are NIS clients in the NIS+ namespace

You should create a minimum of two servers (one master and one replica) for every domain and at least one replica for every physical location. You do not need a replica for every subnet, unless you have Solaris 1.x NIS clients and you want NIS+ servers in NIS-compatibility mode to support the NIS clients. NIS clients do not have access to servers that are not on the same subnet. The only exception are the Solaris 2.x NIS clients, which can use `ypinit`(1M) to specify a list of NIS servers. The netmask number in these cases would have to be set appropriately.

One way you can have a sufficient number of replicas per domain without using a multiplicity of machines is to create multihomed servers. A multihomed server is a machine with multiple ethernet or network interfaces. A multihomed server can serve multiple subnets in a domain.

If the domain hierarchy that you design spans a wide area network (WAN) link, it would be prudent to replicate the domain on either side of the WAN link—with a master server on one side and a replica on the other. This could possibly enable clients on the other side of the link to continue with NIS+ service even if the WAN link were temporarily disabled. Putting servers on either side of a WAN, however, does change the structure of a namespace that is organized by group function rather than by physical layout, since the replica might physically reside within the geographic perimeter of a different domain.

### *Domains Across Time Zones*

Geographically dispersed organizations may determine that organizing their domain hierarchy by functional groups would cause a domain to span more than one time zone. It is *strongly* recommended that you do *not* have domains that span multiple time zones. If you do need to configure a domain across time zones, be aware that a replica's time will be taken from the master server, so the database updates will be synchronized properly, using Greenwich mean time (GMT). This may cause problems if the replica machine is used for other services that are time critical. To make domains across time zones work, the replica's `/etc/TIMEZONE` file has to be locally set to the master server's time zone when you are installing NIS+. Once the replica is running, some time critical programs may run properly and some may not, depending on whether these programs use universal or local time.

### *Information Management*

It is best to use a model of local administration within centralized constraints for managing the information in an NIS+ namespace. Information should be managed, as much as possible, from within its home domain, but according to guidelines or policies set at the global namespace level. This provides the greatest degree of domain independence while maintaining consistency across domains.

## *Domain Names*

Consider name length and complexity. First, choose names that are descriptive. For example, “Sales” is considerably more descriptive than “BW23A.” Second, choose short names. To make your administrative work easier, avoid long names such as “EmployeeAdministrationServices.WizardCorporation”.

A domain name is formed from left to right, starting with the local domain and ending with the root domain.

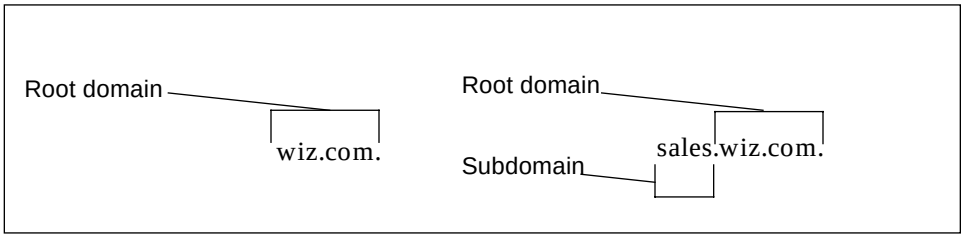



*Figure 2-3* Domain Names Syntax

The root domain must always have at least two labels and must end in a dot. The second label can be an Internet domain name, such as “com.”

Also consider implications of particular names for email domains, both within the company and over the Internet.

Depending on the migration strategy chosen, a viable alternative could be to change domain names on NIS to the desired structure, then migrate to NIS+ domain by domain.

### *The Email Environment*

Because NIS+ can have a domain hierarchy while NIS has a flat domain space, changing to NIS+ can have effects on your mail environment. With NIS, only one mail host is required. If you use a domain hierarchy for NIS+, you may need one mail host for each domain in the namespace because names in separate domains may no longer be unique.

Therefore, the email addresses of clients who are not in the root domain may change. As a general rule, client email addresses can change when domain names change or when new levels are added to the hierarchy.

In earlier Solaris releases, these changes required a great deal of work. This release provided several `sendmail` enhancements to make the task easier. In addition, NIS+ provides a sendmailvars table. The `sendmail` program first looks at the sendmailvars table (see Table 2-1 on page 25), then examines the local `sendmail.cf` file.

---

**Note –** Be sure that mail servers reside in the NIS+ domain whose clients they support. For performance reasons, do not use paths to direct mail servers to tables in other domains.

---

Consider the impact of the new mail addresses on DNS. You may need to adjust the DNS MX records.

## *Selecting the Namespace Servers*

Each NIS+ domain is supported by a set of NIS+ servers. The servers store the domain's directories, groups, and tables, and answer requests for access from users, administrators, and applications. Each domain is supported by only one set of servers. However, a single set of servers can support more than one domain.

Remember that a domain is not an object, but a reference to a collection of objects. Therefore, a server that supports a domain is not actually associated with the domain but with the domain's directories. A domain consists of four directories: `domain`, `ctx_dir.domain`, `org_dir.domain`, and `groups_dir.domain`, as shown in Figure 2-4.

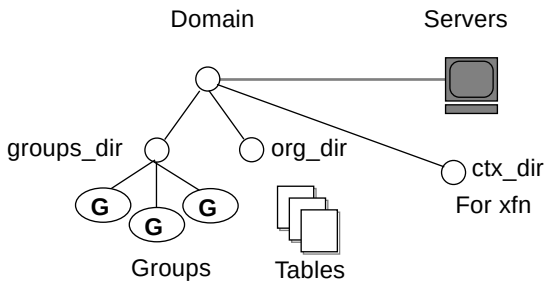



*Figure 2-4* Server Relationship to Domain

Any Solaris 2.x–based workstation can be an NIS+ server as long as it has its own hard disk of sufficient size. The software for both NIS+ servers and clients is included in the Solaris product. Therefore, any workstation that has a Solaris 2.x release installed can become a server or a client, or both.

When selecting the servers that will support the NIS+ namespace, consider the following factors, discussed in the following sections:

- "Supported Domains"
- "Server Load"
- "Disk Space and Memory Requirements"

## *Supported Domains*

When selecting servers, you must differentiate between the requirements imposed by the NIS+ service and those imposed by the traffic load of your namespace.

The NIS+ service requires you to assign at least one server, the master, to each NIS+ domain. (An NIS+ server is capable of supporting more than one domain, but use the one server for one domain configuration in small

namespaces or testing situations.) How many other servers a domain requires is determined by the traffic load, the network configuration, and whether NIS clients are present.

The traffic loads you anticipate will determine the total number of servers used to support the namespace, how much storage and processing speed each will require, and whether a domain needs replicas to ensure its availability. How can you determine how many servers you need?

A good way to begin is by assigning one master server to each domain in the hierarchy.

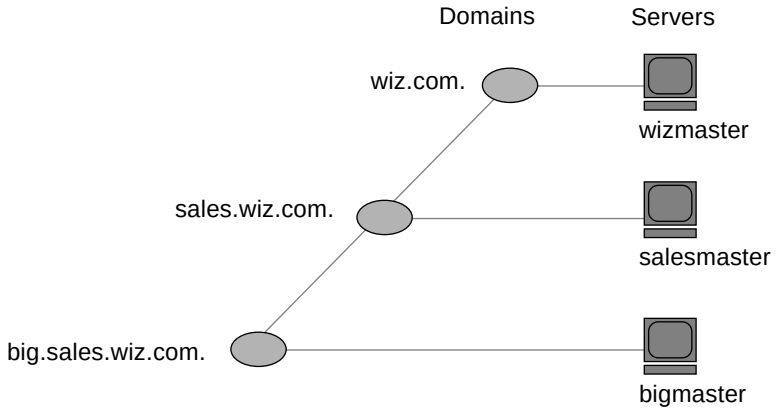



*Figure 2-5* Assigning Servers to Domains

If certain domains must always be available, add two or more replicas to them. Two replicas allow requests to still be answered even if one of the replicas is damaged. Requests may not be answered in a timely manner if a master has only one replica and that replica is being repaired or updated. A domain with

only one replica loses 50 percent of its load capacity when the replica is down. Always add at least one replica to a domain. In small to medium domains, configurations with two to four replicas are normal.

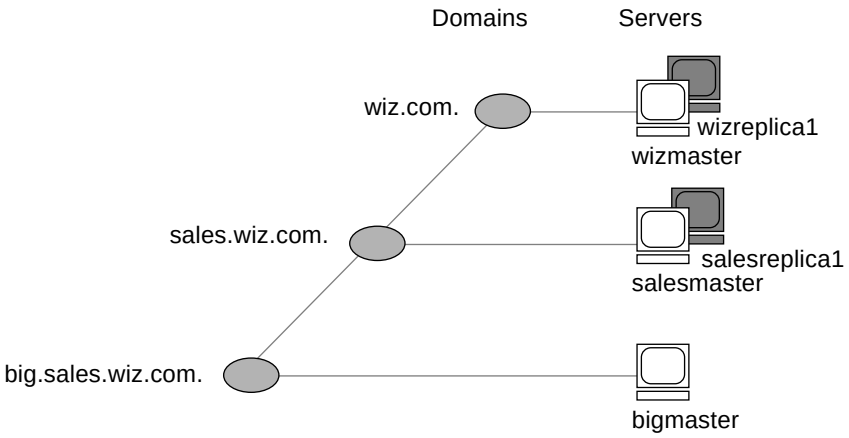



*Figure 2-6* Adding Replicas to a Domain

In organizations with many distributed sites, each site often needs its own subdomain. Typically, the subdomain master is placed in a higher-level domain. As a result, there can be a great deal of traffic between point-to-point links. Creating local replicas can speed request response and minimize point-to-point traffic across the link. In this configuration, lookups may be handled locally.

## *Server Load*

NIS+ master servers require fewer replicas than NIS servers did, since NIS+ does not depend on broadcasts on the local subnet.

Putting replicas on both sides of a weak network link (such as WAN links) is recommended. If the link breaks and the networks are decoupled, both sides of the network can still obtain service.

Do not put more than 10 replicas on one domain. If you can, put one on each subnet; otherwise, distribute the servers as best you can and optimize for the best performance. You don’t need NIS+ servers on every subnet, unless they support NIS clients. In such cases, you may want to install NIS+ servers on multihomed machines.

Try to keep fewer than 1000 clients in a domain. NIS+ clients present a higher load on servers than NIS clients. A large number of clients served by only a few servers may impact network performance.

## *Disk Space and Memory Requirements*

How much disk space you need depends on four factors:

- Disk space consumed by the Solaris 2.x software
- Disk space for `/var/nis` (and `/var/yp`)
- Amount of memory
- Swap space required for NIS+ server processes

The Solaris 2.x software can require over 220 Mbytes of disk space, depending on how much of it you install. For exact numbers, see your Solaris installation guides. You should also count the disk space consumed by other software the server may use. The NIS+ software itself is part of the Solaris 2.x distribution, so it does not consume additional disk space.

NIS+ directories, groups, tables, and client information are stored in `/var/nis`. The `/var/nis` directory uses about 5 Kbytes of disk space per client. For example purposes only, if a namespace has 1000 clients, `/var/nis` requires about 5 Mbytes of disk space. However, because transaction logs (also kept in `/var/nis`) can grow large, you may want additional space per client—an additional 10–15 Mbytes is recommended. In other words, for 1000 clients, allocate 15 to 20 Mbytes for `/var/nis`. You can reduce this if you checkpoint transaction logs regularly. You should create a separate partition for `/var/nis`. This separate partition will help during an operating system upgrade.

If you will use NIS+ concurrently with NIS, allocate space equal to the amount you are allocating to `/var/nis` for `/var/yp` to hold the NIS maps that you transfer from NIS.

Although 32 Mbytes is the minimum memory requirement for servers, it is better to equip servers of medium-to-large domains with at least 64 Mbytes.

You also need swap space equal to three times or more of the size of the NIS+ server process—in addition to the server's normal swap space requirements. The size of the `rpc.nisd` process as shown by the `ps -efl` command. Most of this space is used during callback operations or when directories are

checkpointed (with `nisping -C`) or replicated, because during such procedures an entire NIS+ server process is forked. In no case should you use less than 64 Mbytes of swap space.

## Determine Table Configurations

NIS+ tables provide several features not found in simple text files or maps. They have a column-entry structure, accept search paths, can be linked together, and can be configured in several different ways. You can also create your own custom NIS+ tables. When selecting the table configurations for your domains, consider the following factors discussed in the following sections:

- "Differences Between NIS+ Tables and NIS Maps"
- "Use of Custom NIS+ Tables"
- "Connections Between Tables"

### Differences Between NIS+ Tables and NIS Maps

NIS+ tables differ from NIS maps in many ways, but two of those differences should be kept in mind during your namespace design:

- NIS+ uses fewer standard tables than NIS
- NIS+ tables interoperate with `/etc` files differently than NIS maps did in the SunOS 4.x releases

#### NIS+ Standard Tables

Review the 17 standard NIS+ tables to make sure they suit the needs of your site. They are listed in Table 2-1 on page 25. Table 2-2 on page 26 lists the correspondences between NIS maps and NIS+ tables.

Do not worry about synchronizing related tables. The NIS+ tables store essentially the same information as NIS maps, but they consolidate similar information into a single table (for example, the NIS+ hosts table stores the same information as the `hosts.byaddr` and `hosts.byname` NIS maps). Instead of the key-value pairs used in NIS maps, NIS+ tables use columns and rows. (See *NIS+ and DNS Setup and Configuration Guide*.) Key-value tables have two columns, with the first column being the key and the second column being the value. Therefore, when you update any information, such as host

information, you need only update it in one place, such as the hosts table. You no longer have to worry about keeping that information consistent across related maps.

Note the new names of the automounter tables:

- auto_home (old name: `auto.home`)
- auto_master (old name: `auto.master`)

The dots were changed to underscores because NIS+ uses dots to separate directories. Dots in a table name will cause NIS+ to mistranslate names. For the same reason, machine names cannot contain any dots. You must change any machine name that contains a dot to something else. For example, a machine named `sales.alpha` is not allowed. You could change it to `sales_alpha` or `salesalpha` or any other name that does not contain a dot.

To make the transition from NIS to NIS+, you must change the dots in your NIS automounter maps to underscores. You may also need to do this on your clients' automounter configuration files.

*Table 2-1* NIS+ Tables

| NIS+ Table | Information in the Table |
|---|---|
| hosts | Network address and host name of every workstation in the domain |
| bootparams | Location of the root, swap, and dump partition of every diskless client in the domain |
| passwd | Password information about every user in the domain |
| cred | Credentials for principals who belong to the domain |
| group | The group password, group ID, and members of every UNIX® group in the domain |
| netgroup | The netgroups to which workstations and users in the domain may belong |
| mail_aliases | Information about the mail aliases of users in the domain |
| timezone | The time zone of the domain |
| networks | The networks in the domain and their canonical names |
| netmasks | The networks in the domain and their associated netmasks |
| ethers | The ethernet address of every workstation in the domain |

*Table 2-1* NIS+ Tables *(Continued)*

| NIS+ Table | Information in the Table |
|---|---|
| services | The names of IP services used in the domain and their port numbers |
| protocols | The list of IP protocols used in the domain |
| rpc | The RPC program numbers for RPC services available in the domain |
| auto_home | The location of all user's home directories in the domain |
| auto_master | Automounter map information |
| sendmailvars | Stores the mail domain |

*Table 2-2* Correspondences Between NIS Maps and NIS+ Tables

| NIS Map | NIS+ Table | Notes |
|---|---|---|
| `auto.home` | auto_home | |
| `auto.master` | auto_master | |
| `bootparams` | `bootparams` | |
| `ethers.byaddr` | `ethers` | |
| `ethers.byname` | `ethers` | |
| `group.bygid` | `group` | Not the same as NIS+ groups |
| `group.byname` | group | Not the same as NIS+ groups |
| `hosts.byaddr` | hosts | |
| `hosts.byname` | hosts | |
| `mail.aliases` | mail_aliases | |
| `mail.byaddr` | mail_aliases | |
| `netgroup` | netgroup | |
| `netgroup.byhost` | netgroup | |
| `netgroup.byuser` | netgroup | |
| `netid.byname` | cred | |
| `netmasks.byaddr` | netmasks | |

*Table 2-2* Correspondences Between NIS Maps and NIS+ Tables *(Continued)*

| NIS Map | NIS+ Table | Notes |
|---|---|---|
| `networks.byaddr` | networks | |
| `networks.byname` | networks | |
| `passwd.byname` | passwd | |
| `passwd.byuid` | passwd | |
| `protocols.byname` | protocols | |
| `protocols.bynumber` | protocols | |
| `publickey.byname` | cred | |
| `rpc.bynumber` | rpc | |
| `services.byname` | services | |
| `ypservers` | | Not needed |

NIS+ has one new table for which there is no corresponding NIS table: sendmailvars. The sendmailvars table stores the mail domain used by `sendmail`.

## *NIS+ Tables Interoperate Differently With* `/etc` *Files*

The manner in which NIS and other network information services interacted with `/etc` files in the SunOS 4.x environment was controlled by the `/etc` files using the `+/-` syntax. How NIS+, NIS, DNS and other network information services interact with `/etc` files in the Solaris 2.x release is determined by the *name service switch*. The *switch* is a configuration file, `/etc/nsswitch.conf`, located on every Solaris 2.x client. It specifies the sources of information for

that client: `/etc` files, DNS zone files (hosts only), NIS maps, or NIS+ tables. The `nsswitch.conf` configuration file of NIS+ clients resembles the example in Table 2-3 on page 28.

*Table 2-3* Sample Name Service Switch File

```
passwd:     files
group:      compat
group_compat: nisplus

hosts:      nisplus [NOTFOUND=return] files
services:   nisplus [NOTFOUND=return] files
networks:   nisplus [NOTFOUND=return] files
protocols:  nisplus [NOTFOUND=return] files
rpc:        nisplus [NOTFOUND=return] files
ethers:     nisplus [NOTFOUND=return] files
netmasks:   nisplus [NOTFOUND=return] files
bootparams: nisplus [NOTFOUND=return] files

publickey:  nisplus

netgroup:   nisplus

automount:  files nisplus
aliases:    files nisplus
```

In other words, for most types of information, the source is first an NIS+ table, then an `/etc` file. For the `passwd` and `group` entries, the sources can either be network files or from `/etc` files and NIS+ tables as indicated by +/- entries in the files. (See “NIS+ Data Transfer Commands Changing Information in the Passwd Table” on page 49 for additional information regarding the `passwd` entry in the switch file.)

You can select from three versions of the switch-configuration file or you can create your own. For instructions, see *NIS+ and FNS Administration Guide.*

## *Use of Custom NIS+ Tables*

Determine which nonstandard NIS maps you use and their purpose. Can they be converted to NIS+ or replaced with NIS+ standard maps?

Some applications may rely on NIS maps. Will they still function the same way with NIS+, and can they function correctly in a mixed environment?

To build a custom table in NIS+, use `nistbladm`. Remember that you cannot use dots in the table names.

If you want to use NIS+ to support your custom NIS maps, you should create a key-value table, a table with two columns. The first column is the key and the second column is the value. If you then run the NIS+ servers in NIS-compatibility mode, the NIS clients will not notice any change in functionality.

## *Connections Between Tables*

NIS+ tables contain information only about the resources and services in their home domain. If a client tries to find information that is stored in another domain, the client has to provide the other domain name. You can make this "forwarding" automatic by connecting the local table to the remote table. NIS+ tables can be connected in two different ways:

- Through *paths*
- Through *links*

Paths and links should not be used if you are going to have NIS clients in the NIS+ namespace, because NIS clients are unable to follow the paths or links to find the appropriate information.

### *Paths*

If information in a particular NIS+ table is often requested by clients in other domains, consider establishing a path from the local NIS+ table to the one in the other domain.

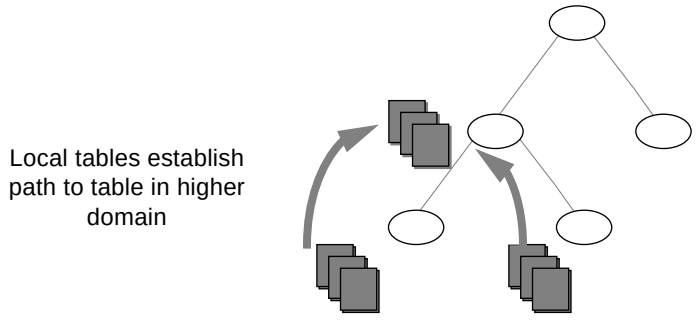



*Figure 2-7* Establishing a Path to Tables in a Higher Domain

Such a path would have two main benefits. First, it would save clients in lower domains the trouble of explicitly searching through a second table. Second, it would allow the administrator in the higher-level domain to make changes in one table and render that change visible to clients in other domains. However, such a path would also hurt performance. Performance is especially affected when searches are unsuccessful, because the NIS+ service must search through two tables instead of one. When you use paths, a table lookup now also depends upon the availability of other domains. This dependence can reduce the net availability of your domain. For these reasons, use paths only if you do not have any other solution to your problem.

You should also be aware that since “mailhost” is often used as an alias, when trying to find information about a specific mail host, you should use its fully qualified name in the search path (for example, mailhost.sales.wiz.com.); otherwise NIS+ will return *all* the “mailhosts” it finds in all the domains it searches through.

The path is established in the local table, with the `-p` option to the `nistbladm` command. To change a table’s path, you must have modify access to the table object. To find out what a table’s search path is, use the `niscat -o` command (you must have read access to the table).

### *Links*

Links between tables produce an effect similar to paths, except that the link involves a search through only one table: the remote table. With a search path, NIS+ first searches the local table, and only if it is unsuccessful does it search the remote table. With a link, the search moves directly to the remote table. In fact, the remote table virtually replaces the local table.

The benefit of a link is that it allows a lower domain to access the information in a higher domain without the need to administer its own table.

To create a link, use the `nisln` command. You must have modify rights to the table object.

Deciding whether to use a path or to link NIS+ tables in a domain is a complex decision, but here are some basic principles:

- Every domain must have access to every standard table.
- Volatile, frequently accessed data should be located lower in the hierarchy. Such data should be located closer to where it is used most often.

- Data that is accessed by several domains should be located higher in the hierarchy, unless the domains need to be independent.
- The lower in the hierarchy you place data, the easier it will be to administer autonomously.
- Only NIS+ clients can see tables connected by paths and links. They cannot be seen by NIS clients.

Figure 2-8 summarizes this principle.

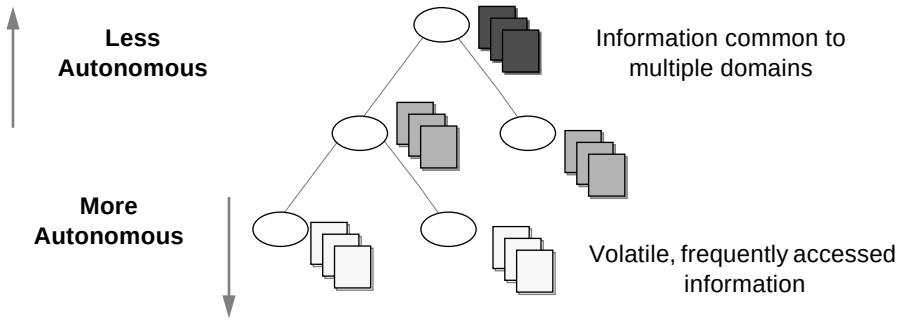



*Figure 2-8* Information Distribution Across an NIS+ Hierarchy

## *Resolving User/Host Name Conflicts*

NIS+ cannot distinguish between a human principal and a workstation principal when requests are made. Therefore, all user names must be different from machine names in the same namespace. In other words, within a given namespace no user can have the same user name as a machine name, and no machine can have the same name as any user ID.

For example, under NIS it was acceptable to have a user with the login name of `irina` whose local machine is also named `irina`. Her network address would be `irina@irina`. This is not allowed under NIS+. When the site is converted to NIS+. either the user will have to change her login name or her machine name will have to be changed. Identical user and machine names are a problem even when the machine with the duplicate name does not belong to the user with the same name. The following examples illustrate duplicate name combinations not valid with NIS+:

- `jane@jane` in the same namespace
- `patna@peshawar` and `rani@patna` in the same namespace

The best solution to this problem is to check all `/etc` files and NIS maps before you use the data to populate NIS+ tables. If you find duplicate names, change the machine names rather than the login names, and later create an alias for the machine's old name.

# *Planning NIS+ Security Measures* 3

This chapter provides general guidelines and recommendations for making choices about security in your namespace.



## *Understanding the Impact of NIS+ Security*

Because NIS+ provides security that NIS did not, it requires more administrative work. It may also require more work from users who are not used to performing `chkey`, `keylogin`, or `keylogout` procedures. Furthermore, the protection provided by NIS+ is not entirely secure. Given enough computing power and the right knowledge, the Diffie-Hellman public-key cryptography system can be broken. In addition, the secret key stored with the key server process is not automatically removed when a credentialed nonroot user logs out unless that user logs out with `keylogout`(1). Security may still be compromised even if the user logs out with `keylogout`(1) because the session keys may remain valid until they expire or are refreshed. (See `keylogout`(1) for more information.) Root's key, created by `keylogin -r`

and stored in `/etc/.rootkey`, remains until the `.rootkey` file is explicitly removed. The superuser cannot use `keylogout`(1). Nevertheless, NIS+ is much more secure than NIS.

## *How NIS+ Security Affects Users*

NIS+ security benefits users because it improves the reliability of the information they obtain from NIS+ and it protects their information from unauthorized access. However, NIS+ security requires users to learn about security and requires them to perform a few extra administrative steps.

Although NIS+ requires a network login, users are not required to perform an additional key login because the `login` command automatically gets the network keys for the client when the client has been correctly configured. Clients are correctly configured when their login password and their Secure RPC password are the same. The secret key for the user `root` is normally made available in the `/etc/.rootkey` file (a possible security problem as noted earlier). When the NIS+ user password and credential are changed with the `passwd` command, the credential information is automatically changed for the user.

- To change the NIS+ machine's local root password, run the `passwd` command.
- To change the root credential, run the `chkey` command.

However, if your site allows users to maintain passwords in their local `/etc/passwd` files in addition to their Secure RPC password's, and if these passwords are different from the Secure RPC password s, then users must run `keylogin` each time they run `login`. The reasons for this are explained in the *NIS+ and FNS Administration Guide.*

## *How NIS+ Security Affects Administrators*

Because the Solaris 2.x release includes the DES encryption mechanism for authentication, administrators who need secure operation do not need to purchase a separate encryption kit. However, administrators must train users how to use the `passwd` and the `passwd -r` commands, and when to use them.

Furthermore, setting up a secure NIS+ namespace is more complex than setting up a namespace without any security. The complexity comes not only from the extra steps required to set up the namespace, but from the job of creating and maintaining user and machine credentials for all NIS+ principals. Administrators have to remove obsolete credentials just as they remove inactive account information from the passwd and hosts tables. Also, when servers' public keys change, administrators have to update the keys throughout the namespace (using `nisupdkeys`). Administrators also have to add LOCAL credentials for users from other domains who want to remote login to this domain and have authenticated access to NIS+.

### *How NIS+ Security Affects Transition Planning*

After you become familiar with the benefits and the administrative requirements of NIS+ security, you must decide whether to implement NIS+ security during or after the transition. It is recommended that you should use full NIS+ security even if you operate some or all servers in a domain in NIS-compatibility mode. (All servers in a domain should have the same NIS-compatibility status.) However, this entails a heavy administrative burden. If you prefer a simpler approach, you could set up the NIS+ servers and namespace with NIS-compatible security, but decline to create credentials for NIS+ clients. Administrators and servers would still require credentials. The NIS+ clients would be relegated to the nobody class, along with the NIS clients. This reduces training and setup requirements, but it has the following drawbacks:

- Users lose the ability to update any NIS+ tables (but they retain their ability to change their login password).
- Users will not be able to verify that the name service information is coming from an authenticated NIS+ server.

## *Selecting Credentials*

NIS+ provides two types of credential: LOCAL and DES. All NIS+ principals need at least one of these credentials. When the namespace is running at security level 2 (the default), all NIS+ principals (clients) must have DES credentials in their home domains. In addition, all users (not workstations) must have LOCAL credentials in their home domains and in every other domain for which they need login access.

To determine the credential needs of your namespace, consider the

- Type of principal
- Type of credential

NIS+ principals can be users or the superuser identity on the client workstation.

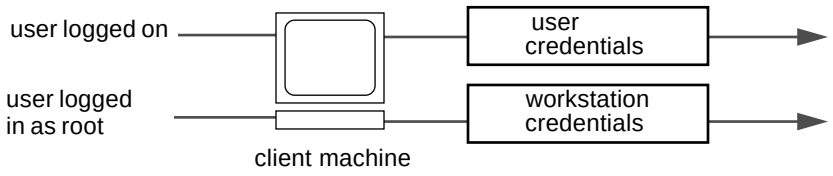



*Figure 3-1* NIS+ Principals

When you determine the credentials you need to create, make sure you know which type of principal the credential is for. For instance, when you set up an NIS+ client with the `nisclient` script, you will create credentials for both the workstation and for the user. Unless credentials for the user are also created, the user would only have the access rights granted to the nobody class. This can work perfectly well. But if you don't give any access rights to the nobody class, the namespace won't be available to users.

## *Choosing a Security Level*

NIS+ is designed to be run at security level 2, which is the default. Security levels 0 and 1 are provided only for the purpose of testing and debugging. Do not run an operational network with real users at any level other than level 2. See *NIS+ and FNS Administration Guide* for more information on the NIS+ security levels.

## *Establishing Password-aging Criteria, Principles, and Rules*

Password-aging is a mechanism that you can use to force users to periodically change their passwords. Password-aging allows you to:

- Specify the maximum number of days that a password can be used before it has to be changed.
- Specify the minimum number of days that a password has to be in existence before it can be changed.

- Specify that a warning message be displayed whenever a user logs in a specified number of days before the user's password time limit is reached.
- Specify the maximum number of days that an account can be inactive. If that number of days pass without the user logging in to the account, the user's password will be locked.

Keep in mind that users who are already logged in when the various maximums or dates are reached are not affected by the above features. They can continue to work as normal. Password-aging limitations and activities are activated only when a user logs in or performs one of the following operations:

- `login`
- `rlogin`
- `telnet`
- `ftp`

These password-aging parameters are applied on a user-by-user basis, and you can have different password-aging requirements for different users. You can also set general default password-aging parameters that apply to all users except those you have individually set.

When planning your NIS+ namespace, decide which password-aging features you want to implement, and the default values you want to specify. For additional information on password-aging, see the Password chapter of *NIS+ and FNS Administration Guide*.

## *Planning NIS+ Groups*

NIS+ introduces a new type of group to name service administration that NIS does not have. An NIS+ group is used only as a means to provide NIS+ access rights to several NIS+ principals at one time; it is used only for NIS+ authorization.

An NIS+ group is one of the four authorization classes on which access rights are based. The four classes are:

- *Owner*. Every NIS+ object has one owner who is a single user. The owner is usually the person who created the object, but ownership can be transferred to another user.
- *Group*. A collection of users grouped together under a group name for the purpose of granting that collection of users specified NIS+ access rights.

- *World.* All *authenticated* users. In other words, any user with valid DES credentials. By definition, an object's owner and members of an object's group are also part of the world class so long as their credentials are valid.
- *Nobody.* Anyone who does not have a valid DES credential. If the credentials of some member of one of the other classes are invalid or missing or corrupted or not found, than that user is placed in the nobody class.

The default name of the group created by NIS+ scripts for such purposes is the *admin* group. You can create other groups with different names, and assign different groups to different NIS+ objects.

Member users of an object's group usually have special privileges to that object, such as permission to make certain changes to the object. For example, you could add several junior administrators to the admin group so that they can only modify the passwd and hosts tables, but they would be unable to modify any other tables. By using an admin group, you can distribute administration tasks across many users and not just reserve them for the superuser of the entire hierarchy. The NIS+ admin group must have credentials created for its members even if you are running the domain in NIS-compatibility mode, because only authenticated users have permission to modify NIS+ tables.

After identifying the type of credentials you need, you should select the access rights that are required in the namespace. To make that task easier, you should first decide how many administrative groups you will need. Using separate groups is useful when you want to assign them different rights. Usually, you create groups by domain. Each domain should have only one admin group.

## *Planning Access Rights to NIS+ Groups and Directories*

After arranging your principals into groups, determine the kind of access rights granted by the objects in the namespace to those groups, as well as to the other classes of principal (nobody, owner, group, and world). Planning these assignments ahead of time will help you establish a coherent security policy.

As shown in Table 3-1, NIS+ provides different default access rights for different namespace objects.

*Table 3-1* Default Access Rights for NIS+ Objects

| Object | Nobody | Owner | Group | World |
|---|---|---|---|---|
| Root-directory object | `r---` | `rmcd` | `rmcd` | `r---` |
| Non-root directory object | `r---` | `rmcd` | `rmcd` | `r---` |
| `groups_dir` directory objects | `r---` | `rmcd` | `rmcd` | `r---` |
| `org_dir` directory objects | `r---` | `rmcd` | `rmcd` | `r---` |
| NIS+ groups | `----` | `rmcd` | `r---` | `r---` |
| NIS+ tables (see Table 3-2 on page 41) | *varies* | *varies* | *varies* | *varies* |

You can use the default rights or assign your own. If you assign your own, you will have to consider how the objects in your namespace will be accessed. Keep in mind that the nobody class comprises all requests from NIS+ clients, whether authenticated or not. The world class comprises all authenticated requests from NIS+ clients. Therefore, if you don't want to provide namespace access to unauthenticated requests, don't assign any access rights to the nobody class; reserve them only for the world class. On the other hand, if you expect some clients—through applications, for instance—to make unauthenticated read requests, you should assign read rights to the nobody class. If you want to support NIS clients in NIS-compatibility mode, you will have to assign read rights to the nobody class.

Also consider the rights each type of namespace object will assign to the NIS+ groups you specified earlier. Depending on how you plan to administer the namespace, you can assign all or some of the available access rights to the group. A good solution is to have the user root on the master server be the owner of the admin group. The admin group should have create and destroy rights on the objects in the root domain. If you want only one administrator to create and modify the root domain, then put just that administrator in the admin group. You can always add additional members to the group. If several administrators may be involved in the setup process, put them all in the group and assign full rights to it. That is easier than switching ownership back and forth.

Finally, the owner of an object should have full rights, although this is not as important if the group does. A namespace is more secure if you give only the owner full rights, but it is easier to administer if you give the administrative group full rights.

## *Planning Access Rights to NIS+ Tables*

NIS+ objects other than NIS+ tables are primarily structural. NIS+ tables, however, are a different kind of object: they are informational. Access to NIS+ tables is required by all NIS+ principals and applications running on behalf of those principals. Therefore, their access requirements are a somewhat different.

Table 3-2 on page 41 lists the default access rights assigned to NIS+ tables. If any columns provide rights in addition to those of the table, they are also listed. You can change these rights at the table and entry level with the `nischmod` command, and at the column level with the `nistbladm -u` command. “Protecting the Encrypted Passwd Field” on page 42 provides just one example of how to change table rights to accommodate different needs.

*Table 3-2* Default Access Rights for NIS+ Tables and Columns

| Table/Column | Nobody | Owner | Group | World |
|---|---|---|---|---|
| hosts table | r--- | rmcd | rmcd | r--- |
| bootparams table | r--- | rmcd | rmcd | r--- |
| passwd table | ---- | rmcd | rmcd | r--- |
| name column | r--- | ---- | ---- | ---- |
| passwd column | ---- | -m-- | ---- | ---- |
| uid column | r--- | ---- | ---- | ---- |
| gid column | r--- | ---- | ---- | ---- |
| gcos column | r--- | -m-- | ---- | ---- |
| home column | r--- | ---- | ---- | ---- |
| shell column | r--- | ---- | ---- | ---- |
| shadow column | ---- | ---- | ---- | ---- |
| group table | ---- | rmcd | rmcd | r--- |
| name column | r--- | ---- | ---- | ---- |
| passwd column | ---- | -m-- | ---- | ---- |
| gid column | r--- | ---- | ---- | ---- |
| members column | r--- | -m-- | ---- | ---- |
| cred table | r--- | rmcd | rmcd | r--- |
| cname column | ---- | ---- | ---- | ---- |
| auth_type column | ---- | ---- | ---- | ---- |
| auth_name column | ---- | ---- | ---- | ---- |
| public_data column | ---- | -m-- | ---- | ---- |
| private_data column | ---- | -m-- | ---- | ---- |
| networks table | r--- | rmcd | rmcd | r--- |
| netmasks table | r--- | rmcd | rmcd | r--- |
| ethers table | r--- | rmcd | rmcd | r--- |
| services table | r--- | rmcd | rmcd | r--- |
| protocols table | r--- | rmcd | rmcd | r--- |

*Table 3-2* Default Access Rights for NIS+ Tables and Columns *(Continued)*

| Table/Column | Nobody | Owner | Group | World |
|---|---|---|---|---|
| rpc table | r--- | rmcd | rmcd | r--- |
| auto_home table | r--- | rmcd | rmcd | r--- |
| auto_master table | r--- | rmcd | rmcd | r--- |

**Note –** NIS-compatible domains give the nobody class read rights to the passwd table at the table level.

## *Protecting the Encrypted Passwd Field*

As you can see in Table 3-2, by default read access is provided to the nobody class by all tables except the passwd table. NIS+ tables give the nobody class read access because many applications that need to access NIS+ tables run as unauthenticated clients. However, if this were also done for the passwd table, it would expose the encrypted passwd column to unauthenticated clients.

The configuration shown in Table 3-2 is the default set of access rights for NIS-compatible domains. NIS-compatible domains must give the nobody class read access to the passwd column because NIS clients are unauthenticated and would otherwise be unable to access their passwd column. Therefore, in an NIS-compatible domain, even though passwords are encrypted, they are vulnerable to decoding. They would be much more secure if they were not readable by anyone except their owner.

Standard NIS+ domains (not NIS-compatible) provide that extra level of security. The default configuration (provided by `nissetup`) uses a column-based scheme to hide the passwd column from unauthenticated users while still providing access to the rest of the passwd table. At the table level, no unauthenticated principals have read access. At the column level, they have read access to every column except the passwd column.

How does an entry owner get access to the passwd column? Entry owners have both read and modify access to their own entries. They obtain read access by being a member of the world class. (Remember that at the table level, the world class has read rights.) They obtain modify access by explicit assignment at the column level.

Keep in mind that table owners and entry owners are rarely and not necessarily the same NIS+ principals. Thus, table-level read access for the owner does not imply read access for the owner of any particular entry.

As mentioned earlier, this is the default setup from the Solaris 2.3 release forward. For a more complete explanation and discussion of table-, entry-, and column level-security, see *NIS+ and FNS Administration Guide*.

# 3

# *Using NIS-Compatibility Mode* 4

Deciding whether and how to run NIS+ in parallel to NIS—and when to stop—is probably one of the most difficult transition issues you will face. NIS+ provides several features that allow it to operate in parallel with NIS; notably, the NIS-compatibility mode.

If you plan to use NIS-compatibility mode, you have to consider:



The essential benefit provided by NIS-compatibility mode is that it does not require you to make any changes to NIS clients. The essential drawback is that you will not be able to take advantage of full NIS+ security and hierarchy and you may have to change those clients' domain names.

Figure 4-1 on page 46 illustrates how you convert from an NIS-only namespace to an NIS-compatible namespace that responds to both NIS and NIS+ requests.

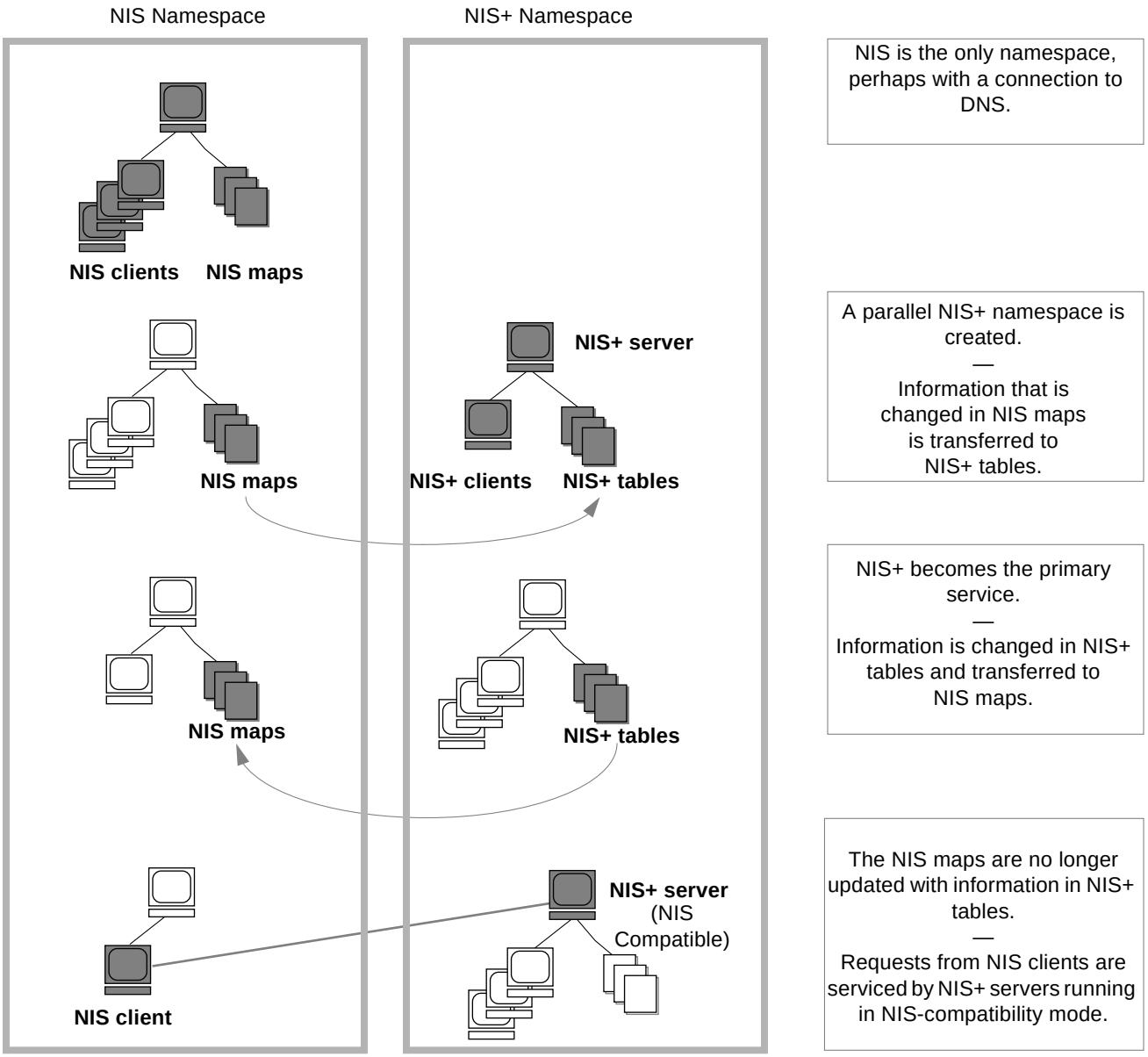



*Figure 4-1* Transition to NIS-Compatibility Mode

## Selecting Your NIS-Compatible Domains

Make a list of your NIS clients and group them in their eventual NIS+ domains. If the NIS+ domain running in NIS-compatibility mode does not have the same name as its NIS clients' original NIS domain, you must change the NIS clients' domain name to the NIS+ domain name that is being supported by the NIS-compatible NIS+ server.

At first, NIS will no doubt be the primary service. As you become familiar with the intricacies of sharing information, you will be able to plan a transition to make NIS+ the primary service.

Some NIS+ users may want the capability of switching back and forth between the main NIS domain and the new NIS+ domain. The `nisclient` script can help them do this easily when backup files are made.

## Determining NIS-Compatible Server Configuration

Take stock of your NIS servers, keeping in mind the requirements for your NIS+ servers. If you plan to eventually use them for the NIS+ service, upgrade them to the NIS+ recommendations. Identify which NIS servers will be used to support which NIS+ domains, and in what capacity (either master or replica). Remember that NIS+ servers belong to the domain *above* the one they support (except for the root domain servers). Since NIS+ servers do not belong to the domain they serve, you will not be able to use the same machines for other services that require domain-dependent information.

If possible, plan to use your NIS+ server machines only for NIS+. This arrangement could require you to transfer other network services, such as DNS name services, boot server, home directories, NFS servers, and so on, to non-NIS+ server machines.

At many sites, the NIS server plays multiple roles, such as NFS server, compute server, `rlogin` server, and mail host server. Because the NIS server uses the same information to resolve its names as do its clients, the NIS server can provide other services as well. As discussed in "The Domain Hierarchy" on page 12, except for root domains, all NIS+ servers live in the domain above the ones that they serve. So either do not run services on an NIS+ server that require access to the name service, or use other means, such as files in `nsswitch.conf`, to acquire this same information. This problem would be solved if there were no hierarchy, the NIS+ root servers would live in the

domain that they serve. The resource requirements of an NIS+ server are greater than those of an NIS server; therefore, it is advisable to not run other services along with NIS+.

If you have non-Solaris machines on your network, then either you can continue to run your NIS+ servers in NIS-compatibility mode or you can move all such machines into their own domain. If you move all non-Solaris machines to one subnet, you can remove the restriction of having NIS+ servers on the same subnet as their NIS-compatible clients require. This will reduce the number of replicas required for any domain.

## *Deciding How to Transfer Information Between Services*

To keep information synchronized, be sure to make one namespace subordinate to the other. At first, the NIS namespace may be the dominant one, in which case you would make changes to the NIS maps and load them into the NIS+ tables. In effect, the NIS namespace would be the master database.

An NIS+ server in NIS-compatibility mode supports standard NIS maps. An exhaustive list of these maps is in the Notes section of the `ypfiles`(4) man page. But there are some limitations on map support: The NIS+ server serves `ypmatch` requests only on the netgroup map, and not on the reverse maps. It does not support enumeration requests (for example, `ypcat`). The `passwd.adjunct` map is not supported, either.

Eventually, the NIS+ namespace should be dominant. When that is the case, you would make changes in the NIS+ tables and copy them to the NIS maps.

The NIS+ `nisaddent` command and the NIS+ `nispopulate` script transfer information between NIS maps and NIS+ tables, as summarized in Table 4-1.

*Table 4-1* NIS+ Data Transfer Commands Changing Information in the Passwd Table

| NIS+ Command | Description |
|---|---|
| `/usr/lib/nis/nisaddent -y` | Transfers information from an NIS map to an NIS+ table after you run `ypxfr` to transfer maps from an NIS server to the local disk. Nonstandard NIS maps can be transferred to NIS+ tables if the information is in key-value pairs. Multicolumned maps will be not be transferred. |
| `/usr/lib/nis/nisaddent -d` | Copies information from an NIS+ table to a file, which can then be transferred to an NIS map with standard NIS utilities. |
| `/usr/lib/nis/nispopulate -Y` | Transfers information from NIS maps to NIS+ tables. |

In versions of NIS+ previous to the Solaris 2.5 release it was necessary to use separate password commands (`passwd`, `yppasswd`, `nispasswd`) to handle password matters, depending on whether a user's password information was stored in `/etc` files, NIS maps, or NIS+ tables. Starting with the Solaris 2.5 release, all of these matters are handled automatically by the `passwd` or `passwd -r nisplus` commands and are controlled by the `passwd` entry in the user's `nsswitch.conf` file.

In order to properly implement the `passwd` command and password aging on your NIS+ or NIS-compatible network, the `passwd` entry of the `nsswitch.conf` file on every machine must be correct. This entry determines where the `passwd` command will go for password information and where it will update password information.

Only five `passwd` entry configurations are permitted:

*Table 4-2* Permitted `passwd nsswitch.conf` Entries

```
passwd: files
passwd: files nis
passwd: files nisplus
passwd: compat
passwd_compat: nisplus
```

**Caution –** All of the `nsswitch.conf` files on all of your network's workstations must use one of the `passwd` configurations shown above. If you configure the `passwd` entry in any other way, users may not be able to log in.

In domains created with NIS-compatibility mode, the permissions are slightly different: permissions at the table level must be set to provide read rights to the world class, and at the column level permissions must provide read access to the nobody class.

## *Deciding How to Implement DNS Forwarding*

NIS servers can forward DNS requests made from Solaris 1.x NIS clients. NIS+ servers running in NIS-compatibility mode also provide DNS forwarding, but only starting with the Solaris 2.3 or later releases. (This feature is available in the Solaris 2.2 release through a patch.) As a result, Solaris 2.x NIS clients must have appropriate `/etc/nsswitch.conf` and `/etc/resolv.conf` files installed locally.

Solaris 1.x NIS clients being supported by Solaris 2.0 or 2.1 servers running in NIS-compatibility mode will not be able to take advantage of DNS forwarding. You must upgrade those servers to Solaris 2.3 or later releases.

If the DNS domains are repartitioned, you must redefine new DNS zone files. Clients, however, may require updates to their `/etc/resolv.conf` file. A client, if it is also a DNS client, can set up its name service switch configuration file to search for host information in either DNS zone files or NIS maps—in addition to NIS+ tables.

### *DNS Forwarding for NIS+ Clients*

NIS+ clients do *not* have implicit DNS-forwarding capabilities like NIS clients do. Instead, they take advantage of the name service switch. To provide DNS capabilities to an NIS+ client, change its hosts entry to:

```
hosts: nisplus dns [NOTFOUND=return] files
```

### *DNS Forwarding for Solaris 2.x NIS Clients*

If an NIS client is using the DNS forwarding capability of an NIS-compatible NIS+ server, the client's `nsswitch.conf` file should *not* have the following syntax in the hosts file:

```
hosts: nis dns files # not for NIS clients
```

Since DNS-forwarding automatically forwards host requests to DNS, the syntax shown above would cause both the NIS+ server and the name service switch to forward unsuccessful requests to the DNS servers, slowing performance.

## *NIS and NIS+ Command Equivalents in the Solaris 1.x and 2.x Releases*

The tables in this section give a quick overview of the differences between Solaris 1.x NIS commands, Solaris 2.x NIS commands, and their NIS+ equivalents.

- Table 4-3 on page 52 describes which NIS commands are supported in the Solaris 2.x release.
- Table 4-4 on page 53 and Table 4-5 on page 54 describe the NIS+ equivalents to NIS client and server commands in the Solaris 2.x release.
- Table 4-6 on page 55 contains a list of the NIS application programming interface functions and their NIS+ API equivalents, if they exist. See the appropriate man pages for details.

### *NIS Commands Supported in the Solaris 2.x Release*

Only some NIS commands are supported in the Solaris 2.x release. NIS server commands are not shipped with the Solaris 2.x release. Just the NIS client commands are included. Whether these NIS commands run also depends on whether a Solaris 2.x NIS client is making requests of an NIS server or of an NIS+ server in NIS-compatibility mode. NIS clients cannot make updates to

NIS+ servers that are running in NIS-compatibility mode. For example, such clients cannot run the `chkey` and `newkey` commands. Table 4-3 lists the NIS commands supported in the Solaris 2.x release.

*Table 4-3* NIS Commands Supported in the Solaris 2.x Releases

| Command Type | NIS Commands Supported in the Solaris 2.x Release | NIS Commands Not Supported in the Solaris 2.x Release |
|---|---|---|
| **Utilities** | `ypinit`<br>`ypxfr`<br>`ypcat`<br>`ypmatch`<br>`yppasswd`<br>`ypset`<br>`ypwhich` | `yppush`<br>`yppoll`<br>`ypchsh`<br>`ypchfn`<br>`ypmake` |
| **Daemons** | `ypbind` | `ypserv`<br>`ypxfrd`<br>`rpc.ypupdated`<br>`rpc.yppasswdd` |
| **NIS API** | `yp_get_default_domain()`<br>`yp_bind()`<br>`yp_unbind()`<br>`yp_match()`<br>`yp_first()`<br>`yp_next()`<br>`yp_all()`<br>`yp_master()`<br>`yperr_string()`<br>`ypprot_err()` | `yp_order()`<br>`yp_update()` |

## *Client and Server Command Equivalents*

The two tables in this section contain NIS commands and their approximate NIS+ equivalents. The commands have been divided into two categories: Table 4-4 on page 53 contains name service client commands and Table 4-5 on page 54 contains name service server commands.

## *Client Command Equivalents*

Table 4-4 on page 53 shows client-to-name server commands. These commands are typed on name service client machines and request information of name service servers. The commands in column 1 will run on Solaris 1.x or 2.x NIS clients connected to Solaris 1.x NIS servers. The commands in column 2 will run on Solaris 1.x or 2.x NIS clients connected to Solaris 2.x NIS+ servers running in NIS-compatibility mode. The commands in column 3 will only run on Solaris 2.x NIS+ clients connected to Solaris 2.x NIS+ servers. Commands are approximately equivalent across rows. "N/A" indicates that an equivalent command does not exist for that case.

*Table 4-4* NIS Client Commands and Equivalent NIS+ Commands

| SunOS 4.x NIS Server | NIS+ Server in NIS- Compatibility Mode | NIS+ Server |
|---|---|---|
| `ypwhich -m` | `ypwhich -m` | `niscat -o` |
| `ypcat` | `ypcat` | `niscat` |
| `ypwhich` | `ypwhich` | N/A |
| `ypmatch` | `ypmatch` | `nismatch/nisgrep` |
| `yppasswd` | `passwd` | `passwd` |
| `ypbind` | `ypbind` | N/A |
| `yppoll` | N/A | N/A |
| `ypset` | `ypset` | N/A |
| N/A | `ypinit -c` | `nisclient -c` |

Note that:

- In the Solaris 2.5 release, the `passwd` command should be used regardless of NIS or NIS+ status. The functions previously performed by `nispasswd` and `yppasswd` have now been included in the `passwd` command.
- The `ypinit -c` command is available only on Solaris 2.x NIS clients.
- The `ypcat` command is not supported for queries directed to the netgroup table. The NIS client's request will time out before an answer is received because this table's format is so different from the `netgroup` NIS map's format.

## *Server Command Equivalents*

Table 4-5 on page 54 shows name server-to-name server commands. The NIS server commands are not included in the Solaris 2.x release, so they are not available to either NIS+ servers or NIS+ servers in NIS-compatibility mode. In addition, an NIS server cannot make updates to an NIS+ server, nor can an NIS+ server make updates to an NIS server. Column 3 lists the NIS+ server commands that are equivalent to the NIS server commands in column 1. There are no exact equivalents for servers in NIS-compatibility mode because NIS-compatibility mode refers only to client commands.

*Table 4-5* NIS Server Commands and Equivalent NIS+ Commands

| SunOS 4.x NIS Server | NIS+ Server in NIS-Compatibility Mode | NIS+ Server |
|---|---|---|
| `ypxfr` | N/A | N/A |
| `makedbm` | N/A | `nisaddent` |
| `ypinit -m`<br>`ypinit -s` | N/A | `nisserver` |
| `ypserv` | `rpc.nisd -Y` | `rpc.nisd` |
| `ypserv -d` | `rpc.nisd -Y -B` | no DNS forwarding needed; use `/etc/nsswitch.conf` |
| `ypxfrd` | N/A | N/A |
| `rpc.ypupdated` | N/A | N/A |
| `rpc.yppasswd` | `rpc.nispasswdd` | `rpc.nispasswdd` |

*Table 4-5 (Continued)*NIS Server Commands and Equivalent NIS+ Commands

| SunOS 4.x NIS Server | NIS+ Server in NIS-Compatibility Mode | NIS+ Server |
|---|---|---|
| `yppush` | N/A | `nisping` |
| `ypmake` | N/A | `nissetup, nisaddent` |
| `ypxfr` | N/A | N/A |

## *NIS and NIS+ API Function Equivalents*

To completely convert your site to NIS+, you must both change the name service and port all applications to NIS+. Any internally created applications that make NIS calls have to be modified to use NIS+ calls. Otherwise, you will always have to run your NIS+ servers in NIS-compatibility mode, with all the drawbacks that this mode entails. External applications may force you to run your namespace in NIS-compatibility mode until they are updated as well.

Table 4-6 contains a list of the NIS API functions and their NIS+ API equivalents, if they exist.

*Table 4-6* NIS API and NIS+ API Equivalent Functions

| NIS API Functions | NIS+ API Functions |
|---|---|
| `yp_get_default_domain ()` | `nis_local_directory()` |
| `ypbind()` | `N/A` |
| `ypunbind()` | `N/A` |
| `ypmatch()` | `nis_list()` |
| `yp_first()` | `nis_first_entry()` |
| `yp_next()` | `nis_next_entry()` |
| `yp_all()` | `nis_list()` |
| `yp_master()` | `nis_lookup()` |
| `yperr_string()` | `nis_perror()` `nis_sperrno()` |

*Table 4-6* NIS API and NIS+ API Equivalent Functions *(Continued)*

| NIS API Functions | NIS+ API Functions |
|---|---|
| `ypprot_err()` | `nis_perror()`<br>`nis_sperrno()` |
| `yp_order()` | `N/A` |
| `yp_update()` | `nis_add_entry()`,<br>`nis_remove_entry()`,<br>`nis_modify_entry()` |

## *NIS-Compatibility Mode Protocol Support*

Table 4-7 shows which NIS protocols are supported by NIS+ servers in NIS-compatibility mode.

*Table 4-7* Support for NIS Protocols by NIS+ Servers

| NIS Protocols | Compatibility Description |
|---|---|
| NIS client V2 protocol | Supported |
| NIS server-to-server protocol | Unsupported |
| NIS client update protocol | `yppasswd` protocol supported |
| NIS client V1 protocol | Not supported except for `YPPROC_NULL`, `YPPROC_DOMAIN`, and `YPPROC_DOMAIN_NONACK` |

# Prerequisites to Transition 5

This chapter describes several miscellaneous tasks that must be carried out before beginning the transition:



## Gauge the Impact of NIS+ on Other Systems

Develop a formal introduction, testing, and familiarization program for your site, not only to train administrators, but also to uncover dependencies of other systems or applications on NIS that will be affected by a transition to NIS+.

For example, some applications may rely on some of the NIS maps. Will they function with standard or custom NIS+ tables? How will their need for access affect your overall security plan?

What nonstandard NIS maps are being used at your site? Can you convert them to NIS+ tables or create nonstandard NIS+ tables to store their information? Be sure to check their access rights.

Does your site use locally built applications that depend on NIS? Do you have commands or applications that make direct NIS calls, such as embedded `yp_match()` function calls? (See "NIS and NIS+ API Function Equivalents" on page 55 for more information.)

Do you have any duplicate user and host names in your namespace? (See "Resolving User/Host Name Conflicts" on page 31 for more information.)

How will the network installation procedures be affected by the transition to NIS+? Analyze the changes required, if any.

Gauging the impact of NIS+ on your site administrative practices will help uncover potential roadblocks.

## *Train Administrators*

Another goal of the introduction and familiarization program discussed in "Become Familiar with NIS+" on page 8 is to give the administrators at your site an opportunity to become familiar with NIS+ concepts and procedures. Classroom instruction alone is insufficient. Administrators need a chance to work in a safe test environment. The training should consist of:

- A formal course in NIS+ concepts and administration.
- Basic NIS+ troubleshooting information and practice.
- Information about your site's implementation strategy and plans.

## *Write a Communications Plan*

Prepare a plan to communicate your intentions to users long before you actually begin converting clients to NIS+. Tell them about the implementation plan and give them a way to obtain more information. As mentioned in Chapter 1, "Introduction," a typical transition goal is to keep the impact of the

transition on clients to a minimum, but users might become concerned about the upcoming change. Send out email notices, conduct informative seminars, and designate email aliases or individuals to whom users can send questions.

## *Identify Required Conversion Tools and Processes*

Consider creating or obtaining transition tools to help with the implementation. If your site already uses automated tools to administer individual systems or network services, consider porting them to operate under the versions of Solaris software and NIS+ that you will be using for the transition (see “Use a Single Release of Software” on page 7). Here are some suggestions for scripts you might want to write:

- A script to convert users to NIS+—make additions to the `nisclient` shell script
- A check script to verify the correctness of a user’s NIS+ environment
- Backup and recovery scripts
- `crontab` entries for routine NIS+ maintenance
- Procedures for notification of outages

Scripts such as these will ensure that the transition is carried out uniformly across domains, speed the transition, and reduce complaints.

You should also prepare a set of standard configuration files and options, such as `nsswitch.conf`, that all clients across the namespace can use.

## *Identify Administrative Groups Used for Transition*

Be sure that the NIS+ groups created as part of your namespace design (see “Establishing Password-aging Criteria, Principles, and Rules” on page 36) correspond to the administrative resources you have identified for the transition. You could require a different set of NIS+ groups for the transition than for routine operation of an NIS+ namespace. Consider adding remote administrators to your groups in case you need their help in an emergency.

Make sure that group members have the proper credentials, that namespace objects grant the proper access rights to groups, and that the right group is identified as the group owner of the right namespace objects.

Table 5-1 provides a summary of commands that operate on NIS+ groups and group permissions useful,

*Table 5-1* NIS+ Commands for Groups

| Command | Description |
|---|---|
| `nisgrpadm` | Creates or deletes groups, adds, changes, lists, or deletes members |
| `niscat -o` | Displays the object properties of an NIS+ group |
| `nissetup` | Creates the basic structure of the directory in which a domain's groups are stored: `groups_dir` |
| `nisls` | Lists the contents of a directory |
| `NIS_GROUP` | Environment variable that overrides the value of `nisdefaults` for the shell in which it is set |
| `nischmod` | Changes an object's access rights |
| `nischown` | Changes the owner of an NIS+ object |
| `nischgrp` | Changes the group owner of an NIS+ object |
| `nistbladm -u` | Changes access rights to NIS+ table columns |
| `nisdefaults` | Displays or changes the current NIS+ defaults |

## *Determine Who Will Own the Domains*

To take complete advantage of the features inherent in a domain hierarchy, distribute the ownership of domains to the organizations they are dedicated to supporting. This will free the administrators of the root domain from performing rudimentary tasks at the local level.

Once you know who owns what, you can provide guidelines for creating administrative groups and setting their access rights to objects.

Consider how to coordinate the ownership of NIS+ domains with the ownership of DNS domains. Here are some guidelines:

- The administration of the DNS domain structure should remain the responsibility of the highest-level administrative group at the site.
- This same administrative group also owns the top-level NIS+ domain.

- Responsibility for the administration of lower-level DNS and NIS+ domains is delegated to individual sites by the top-level administrative group. If the NIS+ domains will be created along the same principles as the DNS domains (for instance, organized geographically), this delegation will be simple to explain.

## *Determine Resource Availability*

Determine what administrative resources will be required for the implementation. These will be above and beyond the resources required for normal operation of NIS+. If your transition will involve a long period of NIS+ and NIS compatibility, additional resources may be required.

Consider not only the implementation of the namespace design but also the conversion the numerous clients and dealing with special requests or problems. Keep in mind that NIS+ has a steep learning curve. Administrators may be less efficient for a while at performing support functions with NIS+ than they were with NIS. Consider not only formal training but extensive lab sessions with hands-on experience.

Finally, even after the transition is complete, administrators will require extra time to become familiar with the everyday work flow of supporting NIS+.

Consider hardware resources. NIS servers are often used to support other network services such as routing, printing, and file management. Because of the potential load on an NIS+ server, you should use dedicated NIS+ servers. This load-balancing simplifies the transition because it makes troubleshooting and performance monitoring simpler. Of course, you incur the cost of additional systems. The question of how many servers you will need and how they should be configured is addressed in Chapter 2, “Designing the NIS+ Namespace.”

Remember, these servers are required in *addition* to the NIS servers. Although the NIS servers might be decommissioned or recycled after the transition is complete, the NIS+ servers will continue to be used.

## *Resolve Conflicts Between Login Names and Host Names*

The NIS+ authentication scheme does not allow workstations and users to use the same names within a domain; for example, joe@joe is not permitted. Since NIS+ does not distinguish between credentials for hosts and login names, you

can only use one credential type per name. If you have duplicate names in your namespace and you must keep the duplicate host name for some other reason make this change: retain the user login name and alias the duplicate host names. Create a new name for the host and use the old name as an alias for the new name. See "Resolving User / Host Name Conflicts" on page 31 for examples of illegal name combinations.

You must resolve name conflicts before the implementation can begin, but you should also plan on permanently checking new workstations and user names during routine NIS+ operation. The `nisclient` script does name comparisons when you use it to create a client credential.

## *Examine All Information Source Files*

Check all `/etc` files and NIS maps for empty fields or corrupted data before configuring NIS+. The NIS+ table-populating scripts and commands might not succeed if the data source files contain empty fields or extraneous characters. Fill blank fields or fix the data before you start. It is better to delete questionable users or machines from the `/etc` files or NIS maps before running NIS+ scripts, and then add them back later after NIS+ is installed, than to proceed with the scripts and possibly corrupt data.

## *Remove the "." from Host Names*

Because NIS+ uses dots (periods) to delimit between machine names and domains and between parent and sub-domains, you cannot have a machine (host) name containing a dot. Before converting to NIS+ you must eliminate any dots in your hostnames. You can convert hostname dots to underscores (_). For example, you cannot have a machine named `sales.alpha`. You can convert that name to `sales_alpha`. (See the `hosts` man page for detailed information on allowable hostnames.)

## *Remove the "." from NIS Map Names*

As described in Chapter 2, "Designing the NIS+ Namespace," NIS+ automounter tables have replaced the "." (dot) in the table name with an underscore. You also need to make this change to the names of NIS maps that you will use during the transition. If you do not, NIS+ will confuse the dot in the name with the periods that distinguish domain levels in object names.

**Note –** Be sure to convert the dot to underscores for *all* NIS maps, not just those of the automounter. Be aware, however, that changing the names of nonstandard NIS maps from dots to underscores may cause applications that use those nonstandard maps to fail unless you also modify the applications to recognize NIS+ syntax.

## *Document Your Current NIS Namespace*

Documenting your current configuration will give you a clear point of departure for the transition. Make a list of the following items:

- Name and location of all current NIS domains and networks
- Host name and location of all current NIS servers, both master and slave
- Configuration of all current NIS servers, including:
  - Host name
  - CPU type
  - Memory size
  - Disk space available
  - Name of administrators with root access
- Nonstandard NIS maps

Correlate the list of your NIS clients with their eventual NIS+ domains. They will have to be upgraded to the Solaris 2.x release.

## *Create a Conversion Plan for Your NIS Servers*

Take stock of your NIS servers. Although you can recycle them after the transition is complete, keep in mind that you will go through a stage in which you will need servers for *both* services. Therefore, you cannot simply plan to satisfy all your NIS+ server needs with your existing NIS servers.

You will find it helpful to create a detailed conversion plan for NIS servers, identifying which NIS servers will be used for NIS+ and when they will be converted. Do not use the NIS servers as NIS+ servers during the first stages of NIS-to-NIS+ transition. As described in Chapter 6, “Implementing the Transition.” the implementation is most stable when you check the operation of the entire namespace as a whole before you convert any clients to NIS+.

# 5

Assign NIS servers to NIS+ domains and identify each server's role (master or replica). Once you have identified the servers you plan to convert to NIS+ service, upgrade them to NIS+ requirements (see "Disk Space and Memory Requirements" on page 23).

# *Implementing the Transition* 6

Once you have performed the tasks described in the previous chapters, most of the hard work is done. Now all you have to do is set up the namespace you designed and add the clients to it. This chapter describes how to do that. Before performing these steps, verify that all your pre-transition tasks have been completed and that users at your site are aware of your plans.

If you will be running NIS+ domains alongside DNS domains, you will set up one NIS+ sub-domain with each DNS domain. After you have set up a complete NIS+ namespace along with the first DNS domain and have verified that everything is working right, then you can set up the other NIS+ namespaces in parallel.

These are the major implementation phases:

# 6

## *Phase I—Set Up the NIS+ Namespace*

Set up the namespace with full DES authentication, even if the domains will operate in NIS-compatibility mode. Use the NIS+ scripts described in *NIS+ and DNS Setup and Configuration Guide* to set up your namespace; see *NIS+ and FNS Administration Guide* for more explanation of NIS+ structure and concepts. Then perform the following steps:

1. **Set up the root domain.**
   If you are going to run the root domain in NIS-compatibility mode, use `nisserver -y`. (If you choose not to use the setup scripts, use the `-Y` flag of `rpc.nisd` and `nissetup`.)

2. **Populate the root domain tables.**
   You can use `nispopulate` to transfer information from NIS maps or text files. Of course, you can also create entries one at a time with `nistbladm` or `nisaddent`.

3. **Set up clients of the root domain.**
   Set up a few clients in the root domain so that you can test its operation properly. Use full DES authentication. Some of these client machines will later be converted to root replica servers and some will serve as workstations for the administrators who support the root domain. NIS+ servers should never be an individual's workstation.

4. **Create or convert site-specific NIS+ tables.**
   If the new NIS+ root domain will require custom, site-specific NIS+ tables, create them, with `nistbladm` and transfer the NIS data into them with `nisaddent`.

5. **Add administrators to root domain groups.**
   Remember, the administrators must have LOCAL and DES credentials (use `nisaddcred`). Their workstations should be root domain clients and their root identities should also be NIS+ clients with DES credentials.

6. **Update the sendmailvars table, if necessary.**
   If your email environment has changed as a result of the new domain structure, populate the root domain's sendmailvars table with the new entries.

7. **Set up root domain replicas.**
   First convert the clients into servers (use `rpc.nisd` with `-Y` for NIS compatibility and use also `-B` if you want DNS forwarding), and then associate the servers with the root domain by running `nisserver -R`.

8. **Test the root domain's operation.**
   Develop a set of installation-specific test routines to verify a client's functioning after the switch to NIS+. This will speed the transition work and reduce complaints. You should operate this domain for about a week before you begin converting other users to NIS+.

9. **Set up the remainder of the namespace.**
   Do not convert any more clients to NIS+, but go ahead and set up all the other domains beneath the root domain. This includes setting up their master and replica servers. Test each new domain as thoroughly as you tested the root domain until you are sure your configurations and scripts work properly.

10. **Test the operation of the namespace.**
    Test all operational procedures for maintenance, backup, recovery, and other scenarios. Test the information-sharing process between all domains in the namespace. Do not proceed to Phase II until the entire NIS+ operational environment has been verified.

11. **Customize the security configuration of the NIS+ domains.**
    This may not be necessary if everything is working well; but if you want to protect some information from unauthorized access, you can change the default permissions of NIS+ tables so that even NIS clients would be unable to access them. You could also rearrange the membership of NIS+ groups and the permissions of NIS+ structural objects to align with administrative responsibilities.

# 6

## *Phase II—Connect the NIS+ Namespace to Other Namespaces*

12. **[Optional] Connect the root domain to the DNS namespace.**
    An NIS+ client can be connected to the Internet using the name service switch. Workstations, if they are also DNS clients, can have their name service switch configuration files set to search for information in DNS zone files—in addition to NIS+ tables or NIS maps.

    Configure each client's `/etc/nsswitch.conf` and `/etc/resolv.conf` files properly. The `/etc/nsswitch.conf` file is the client's name service switch configuration file. The `/etc/resolv.conf` lists the IP addresses of the client's DNS servers; it is described in *NIS+ and DNS Setup and Configuration Guide*.

13. **Test the joint operation of NIS+ with DNS.**
    Verify that requests for information can pass between the namespaces without difficulty.

14. **If operating NIS+ in parallel with NIS, test the transfer of information.**
    Use the `nispopulate` script to transfer information from NIS to NIS+. To transfer data from NIS+ to NIS, run `nisaddent -d` and then `ypmake`. (See the man pages for more information.) Use scripts to automate this process. Establish policies for keeping tables synchronized, particularly the hosts and passwd tables. Test the tools used to maintain consistency between the NIS and NIS+ environments. Decide when to make the NIS+ tables the real source of information.

15. **Test operation of NIS+ with both DNS and NIS.**
    Test all three namespaces together to make sure the added links do not create problems.

## *Phase III—Make the NIS+ Namespace Fully Operational*

16. **Convert clients to NIS+.**
    Convert clients one workgroup at a time, and convert all workgroups in a subnet before starting on those of another subnet. That way, when you convert all the clients in a subnet, you can eliminate the NIS service on that subnet. Run the verification script after converting each client to make sure

that the conversion worked properly. That verification script should inform the user of the support structure that is in place to help with problems and how to report them. The actual steps required depend on the site.

Use the `nisclient` script to convert NIS clients to NIS+ clients. If you need to modify the clients' DNS configuration, you will have to write your own scripts to automate that process.

You can also save time if your site has a shared, mounted central directory similar to `/usr/local`. You could put the script in the central directory and, on the day of conversion, send email to clients asking them to run the script as superuser.

17. **Monitor the status of the transition as clients are being converted.**
    Track progress against your plan and all serious complications not anticipated in the planning stages. Announce your status so that interested parties can track it.

18. **Decommission NIS servers.**
    As all the clients on a subnet are converted to NIS+, decommission the NIS servers. If a particular subnet has some clients that require NIS service, use the NIS-compatibility feature of the NIS+ servers but do not retain the NIS servers.

19. **Evaluate NIS+ performance.**
    Once the implementation is complete, test to see that NIS+ is working correctly.

20. **Optimize the NIS+ environment.**
    Based on the results of your performance evaluation, modify the NIS+ environment as needed. These improvements could be as simple as adding selected replicas in domains with high loads or as involved as rearranging the storage of NIS+ information for a group of domains.

21. **Clean up new domains.**
    If you did not change old domain names during the transition for the sake of simplicity, upgrade them now to the new NIS+ naming scheme. For example, if you left some domains with geographic labels while you converted to an organizational hierarchy, you would change the geographic names to their organizational versions.

# 6

## *Phase IV—Upgrade NIS-Compatible Domains*

22. **Convert the last NIS clients to NIS+.**
    As soon as you can, eliminate the need for NIS-compatible NIS+ domains. Upgrading the last NIS clients to NIS+ will allow you to take advantage of NIS+ security features. You will not be able to eliminate the need for NIS-compatible NIS+ domains if you are running non-Solaris machines on your network.

23. **Adjust your security configuration.**
    Once you have no more NIS clients, you can restart the NIS+ servers in standard mode and run `nischmod` on the NIS+ tables to change permission levels to eliminate the security hole caused by NIS compatibility. If you did not create credentials for NIS+ principals before, you must do that now. Restrict the access of unauthenticated principals.

24. **Establish miscellaneous evaluation and improvement programs.**
    Evaluate operational procedures to determine which ones can be improved, particularly procedures used to recover from problems. Plan for new NIS+ releases and possible functional enhancements. Track the development of Solaris components that might require new NIS+ tables. Look for automated tools that enable you to perform NIS+ administration functions more efficiently. Finally, work with internal developers to help them take advantage of the NIS+ API.

This completes the transition to NIS+.

# *Index*

## Symbols



## A

## B



## C



## D

## E



## F



## G

## H



## I

## K



## L



## M

## N

## O



## P

## Q



## R

## S

## T

## Y

## Z